Ethernet Protocol Stack Chip CH395

Datasheet 1

Version: 2.2

https://wch-ic.com

1. Overview

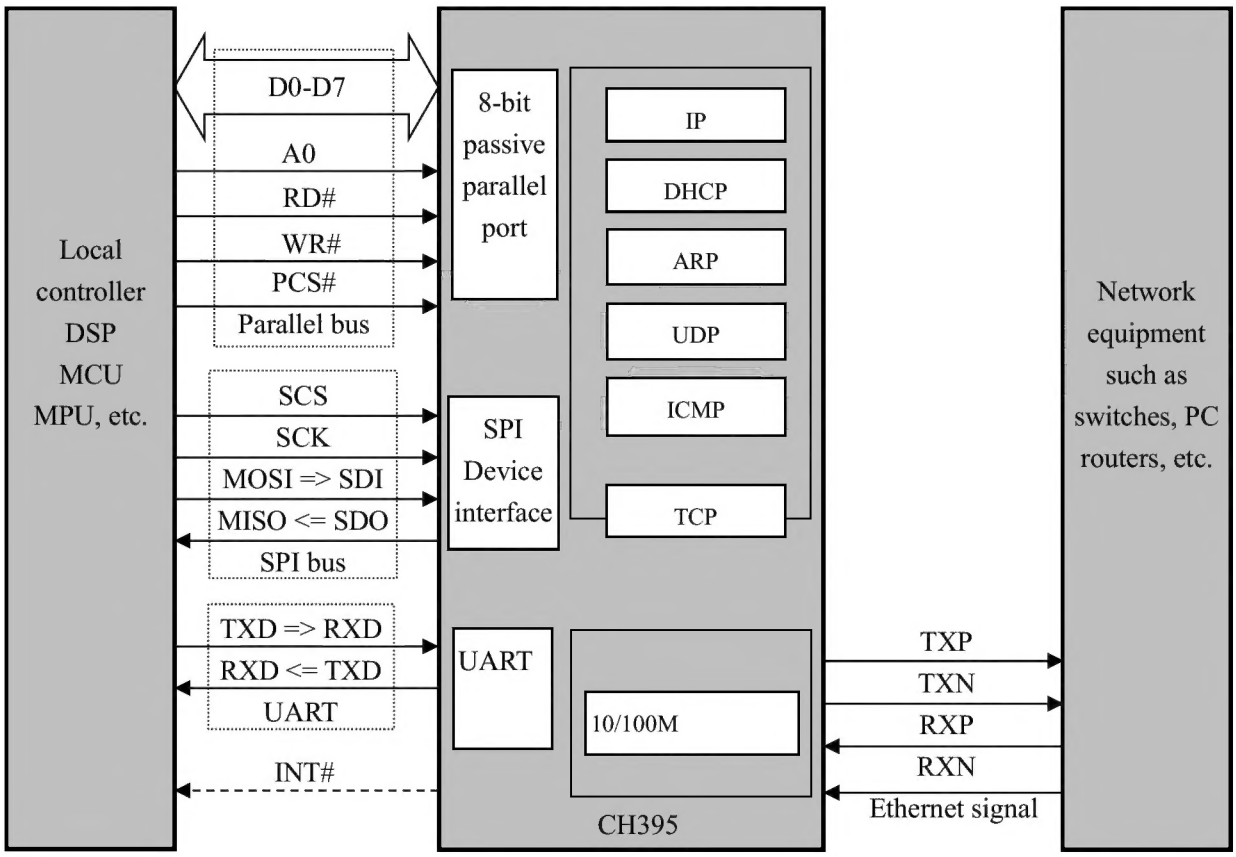
CH395 is an Ethernet protocol stack management chip, which is used for Ethernet communication in MCU system.

CH395 chip comes with 10/100M Ethernet Media Access Control (MAC) and Physical layer (PHY), which is fully compatible with IEEE802.3 protocol, and has built-in Ethernet protocol stack firmware such as IP, ARP, ICMP, UDP and TCP. MCU system can conveniently communicate with the network through CH395 chip.

CH395 supports 3 communication interfaces: 8-bit parallel port, SPI interface and UART. Controllers such as MCU/DSP/MCU/MPU can control the CH395 chip for Ethernet communication through any of the above communication interfaces.

The following figure shows the application block diagram of CH395.

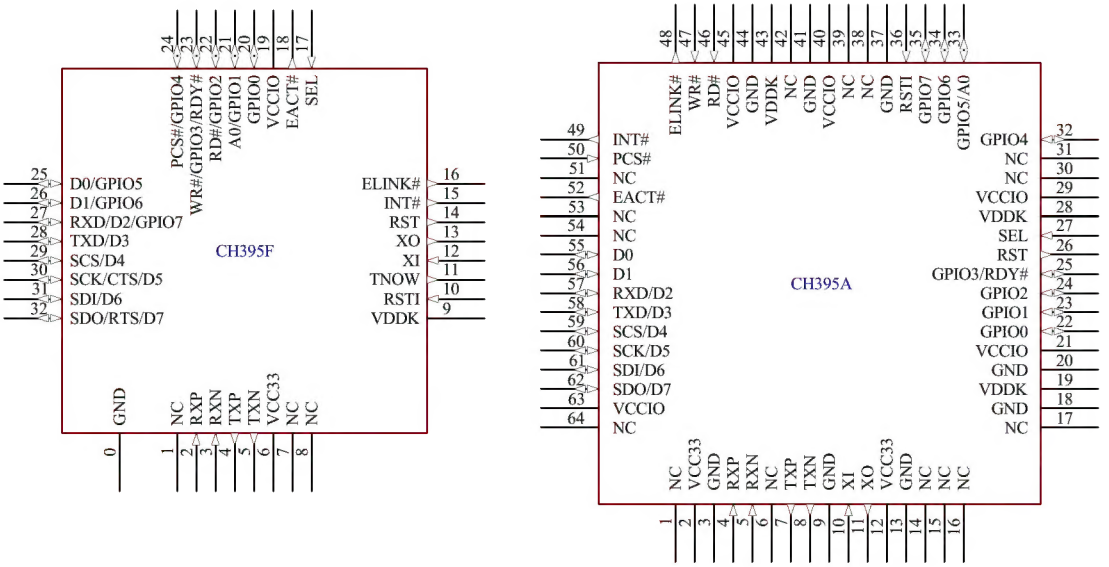
Figure 1-1 CH395 application block diagram



2. Features

- Built-in Ethernet Media Access Control (MAC) and Physical layer (PHY)
- Support 10/100M, full-duplex/half-duplex adaptation, and is compatible with IEEE802.3 protocol
- Support automatic conversion of MDI/MDIX lines
- I/O port supports 3.3V, 2.5V and 1.8V power supply, and is compatible with various voltage standards MCU.
- Built-in network port pull-up resistor, crystal oscillator matching capacitor, simplify peripheral circuits.
- Built-in TCP/IP protocol suite. Support IPv4, ARP, ICMP, UDP and TCP protocols
- Support DHCP to automatically obtain IP address.
- Provide 8 independent Socket pairs, which can transmit and receive data simultaneously.
- Support MACRAW mode and IPRAW mode.
- Provide a SPI slave interface (SPI mode 0 or 3) up to 40MHz, with the high bit coming first
- Provide high-speed 8-bit passive parallel interface and support parallel data bus connected to MCU.
- Provide UART with speeds up to 10Mbps, supports serial port connected to MCU, and supports dynamic adjustment of communication baud rate
- CH395F serial port supports RS485 automatic switching, and supports hardware flow control
- Support Sleep mode
- Built-in 24K RAM can be used for Ethernet data transceiver, and each Socket transceiver buffer can be configured flexibly
- Built-in 4KB EEPROM
- Support 8-channel GPIO
- Provide QFN32, LQFP64M and LQFP128 lead-free packages.

3. Package



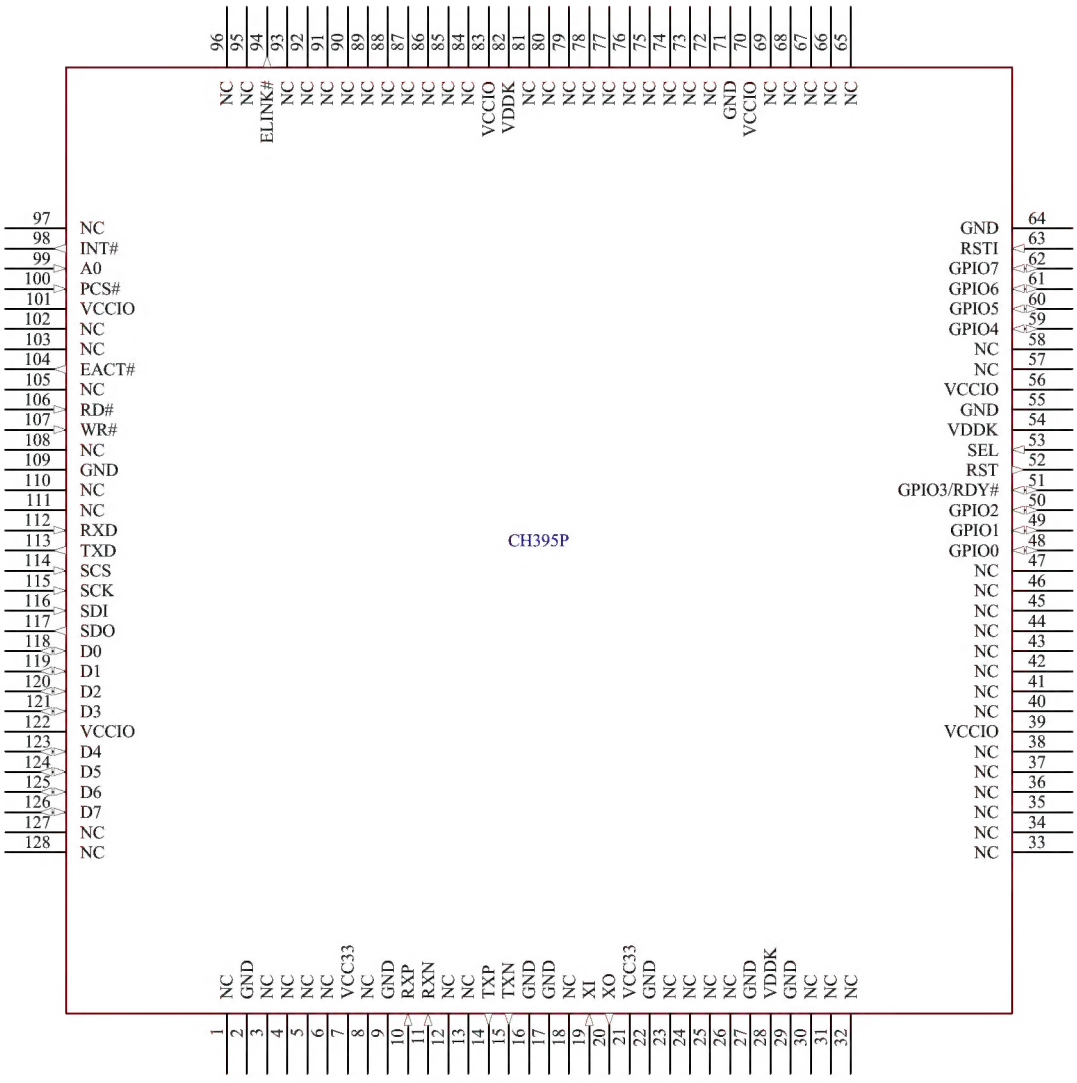

CH395P
NC
INT#
A0
PCS#
VCCIO
NC
NC
EACT#
NC
RD#
WR#
NC
GND
NC
NC
RXD
TXD
SCS
SCK
SDI
SDO
D0
D1
D2
D3
VCCIO
D4
D5
D6
D7
NC
NC
GND
RSTI
GPIO7
GPIO6
GPIO5
GPIO4
NC
NC
VCCIO
GND
VDDK
SEL
RST
GPIO3/RDY#
GPIO2
GPIO1
GPIO0
NC
NC
NC
NC
NC
NC
NC
NC
NC
VCCIO
NC
NC
NC
NC
NC
NC
NC
NC
NC
NC
NC
NC
ELINK#
VCCIO
VDDK
GND
VCCIO
NC
NC
NC
NC
NC
NC
NC
NC
NC
NC
NC
NC

Table 3-1 Package description

Package Form	Body Size	Pin Pitch		Package Description	Order Model
QFN32	4*4mm	0.4mm	15.7mil	Quad Flat No-leads Package	CH395F
LQFP64M	10*10mm	0.5mm	19.7mil	Low-profile Quad Flat Package	CH395A
LQFP128	14*14mm	0.4mm	15.7mil	Low-profile Quad Flat Package	CH395P

Note: 1. It is recommended to use smaller package CH395F for new design.

2 CH395A based on CH395Q upgrade, CH395P based on CH395L upgrade, the pin is basically compatible, replacement needs to adjust the peripheral circuit.

3. CH395Q, CH395L maintain the supply but not recommended for new designs.

4. CH395F Pins

4.1 CH395F Pin Definition

Table 4-1 CH395F pin definition

CH395F Pin No.	Pin Name	Type(1)	Pin Description(2)
2	RXP	I/O	Differential input in 10BASE-T/100BASE-TX MDI mode; Differential output in 10BASE-T/100BASE-TX MDIX mode.
3	RXN		
4	TXP	I/O	Differential output in 10BASE-T/100BASE-TX MDI mode; Differential input in 10BASE-T/100BASE-TX MDIX mode.
5	TXN		
6	VCC33	P	3.3V power supply input, it is recommended to place 0.1uF parallel 10uF or 4.7uF ground capacitor close to the core.
19	VCCIO	P	For the power input of I/O interface, it is recommended to place 0.1uF capacitance to ground close to the chip.
9	VDDK	P	The external 1uF ground capacitor is placed close to the chip.
0	GND	P	Common ground.
1/7/8	NC	-	Reserved, suggested suspension.
12	XI	I	The crystal oscillator input needs an external 25MHz crystal end or an external clock input with a built-in crystal oscillator matching capacitor.
13	XO	O	The inverted output of the crystal oscillator needs to be externally connected to the other end of the 25MHz crystal, and a crystal oscillator matching capacitor is built in.
10	RSTI	I,PU	External reset input, active low, built-in pull-up resistor.
11	TNOW	O	Transmit status output, which is used to control RS485 transceiver switching of serial port.
14	RST	O	Reset and external reset output, active high.
15	INT#	O	Interrupt request output, active low.
16	ELINK#	O	Network connection indicator LED output: Low level indicates that Ethernet PHY is connected; High level indicates that Ethernet PHY is not connected.
17	SEL	I,PU	Used as an interface configuration input during internal chip reset. For details on the specific configuration method, please refer to Chapter 6.1.
18	EACT#	O	Carrier sensing indicator LED output: LED flashes indicate a carrier sensing signal.
20	GPIO0	I/O	General input and output pin 0, configured as input by default.
21	A0/GPIO1	I/O,PU	A0: Address input of parallel port, distinguishing command port from data port, built-in pull-up resistor; When A0=1, you can write commands, and when A0=0, you can read and write data. GPIO1: General-purpose input and output pin 1, configured as input by default.
22	RD#/GPIO2	I/O,PU	RD#: The parallel port read gate input, active low, built-in pull-up resistor.

			GPIO2: General-purpose input and output pin 2, configured as input by default.
23	WR#/GPIO3/RDY#	I/O,PU	WR#: The parallel port write gate input, active low, built-in pull-up resistor. GPIO3: General-purpose input and output pin 3, configured as input by default. RDY#: Output low level after CH395F reset is completed; RDY# function is not effective in parallel port mode.
24	PCS#/GPIO4	I/O,PU	PCS#: Chip selection control signal input of parallel port, active low level, built-in pull-up resistor. GPIO4: General-purpose input and output pin 4, configured as input by default.
25	D0/GPIO5	I/O,PU	D0: The 0th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor. GPIO5: General-purpose input and output pin 5, configured as input by default.
26	D1/GPIO6	I/O,PU	D1: The first bit in the 8-bit bidirectional data bus with a parallel port, with a built-in pull-up resistor. GPIO6: General-purpose input and output pin 6, configured as input by default.
27	RXD/D2/GPIO7	I/O,PU	RXD: Serial data input of UART, built-in pull-up resistor; D2: The second bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor. GPIO7: General-purpose input and output pin 7, configured as input by default.
28	TXD/D3	I/O,PU	During the internal reset of the chip, the interface configuration input is input, and the built-in pull-up resistor is used in Chapter 6.1. After the chip reset is completed, use it as an interface pin. TXD: Serial data output of asynchronous serial port. D3: The third bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
29	SCS/D4	I/O,PU	SCS: Chip-select input SCS of the SPI interface, active low, built-in pull-up resistor. D4: The 4th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
30	SCK/CTS/D5	I/O,PU	SCK: The serial clock input SCK of the SPI interface, built-in pull-up resistor. CTS: Clear the send input, built-in pull-up resistor. If it is high, the serial port blocks the next data transmission when the current data transmission is over. You can connect to the countertop RTS to achieve hardware flow control (the serial port flow control function is turned off by default and can be turned on through the CMD_SET_FUN_PARA command). D5: The 5th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.

31	SDI/D6	I/O,PU	SDI: Serial data input to SDI of the SPI interface, built-in pull-up resistor. D6: The 6th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
32	SDO/RTS/D7	I/O,PU	SDO: Serial data output as SPI interface. RTS: Transmit request output. If it is low, it indicates that the serial port is ready to receive data. You can connect to the countertop CTS to achieve hardware flow control (the serial port flow control function is turned off by default and can be turned on through the CMD_SET_FUN_PARA command). D7: The 7th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.

Note 1: Table abbreviation explanation:

I = Input; O = Output; I/O = Input/Output;

P = Power supply; PU = Built-in pull-up resistor.

Note 2: The specific function of the alternate pin of the CH395F depends on the currently selected interface mode. When the alternate pin is used for the interface function, its GPIO function will be occupied, causing its GPIO function to fail, and at the same time, the corresponding bits in the GPIO register have no practical significance.

4.2 CH395A Pin Definition

Table 4-2 CH395A pin definition

CH395A Pin No.	Pin Name	Type(1)	Pin Description(2)
4	RXP	I/O	Differential input in 10BASE-T/100BASE-TX MDI mode; Differential output in 10BASE-T/100BASE-TX MDIX mode.
5	RXN		
7	TXP	I/O	Differential output in 10BASE-T/100BASE-TX MDI mode; Differential input in 10BASE-T/100BASE-TX MDIX mode.
8	TXN		
2/12	VCC33	P	3.3V power supply input, it is recommended that 0.1uF parallel 10uF or 4.7uF ground capacitor be placed close to the chip.
21/29/40/45/63	VCCIO	P	For the power input of the I/O interface, it is recommended to place the 0.1uF ground capacitor close to the chip.
19	VDDK	P	The external 1uF ground capacitor is placed close to the chip.
28/43	VDDK	P	The external 0.1uF ground capacitor is placed close to the chip.
3/9/13/18/20/37/41/44	GND	P	Common ground.
1/6/14/15/16/17/30/31/38/39/42/51/53/54/64	NC	-	Reserved, suggested suspension.
10	XI	I	The crystal oscillator input needs an external 25MHz crystal end or an external clock input with a built-in crystal oscillator matching capacitor.
11	XO	O	The inverted output of the crystal oscillator needs to be externally connected to the other end of the 25MHz crystal, and a crystal oscillator matching capacitor is built in.
22	GPIO0	I/O	General-purpose input and output pin 0, configured as input by default.
23	GPIO1	I/O	General-purpose input and output pin 1, configured as input by default.
24	GPIO2	I/O	General-purpose input and output pin 2, configured as input by default.
25	GPIO3/RDY#	I/O	GPIO3: General-purpose input and output pin 3, configured as input by default. RDY#: After CH395A is reset, the output is low.
26	RST	O	Power-on reset and external reset output, active high level.
27	SEL	I,PU	Used as an interface configuration input during internal chip reset. For details on the specific configuration method, please refer to Chapter 6.1.
32	GPIO4	I/O	General-purpose input and output pin 4, configured as input by default.
33	GPIO5/A0	I/O,PU	GPIO5: General-purpose input and output pin 5, configured as input by default. A0: The address input of the parallel port is distinguished from the command port and the data port, and the built-in pull-up resistor is used; when A0=1, the command can be written, and when A0=0, the data can be read and written.
34	GPIO6	I/O	General-purpose input and output pin 6, configured as input by default.

35	GPIO7	I/O	General-purpose input and output pin 7, configured as input by default.
36	RSTI	I,PU	External reset input, active low, built-in pull-up resistor.
46	RD#	I,PU	The parallel port read gate input, active low, built-in pull-up resistor.
47	WR#	I,PU	The parallel port write gate input, active low, built-in pull-up resistor.
49	INT#	O	Interrupt request output, active low.
50	PCS#	I,PU	The chip select control input of parallel port, active low, built-in pull-up resistor.
55	D0	I/O,PU	The 0th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
56	D1	I/O,PU	The 1st bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
57	RXD/D2	I/O,PU	RXD: Serial data input of UART, built-in pull-up resistor; D2: The 2nd bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
58	TXD/D3	I/O,PU	During the internal reset of the chip, the interface configuration input is input, and the built-in pull-up resistor is used in Chapter 6.1. After the chip reset is completed, use it as an interface pin. TXD: Serial data output of UART. D3: The 3rd bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
59	SCS/D4	I/O,PU	SCS: Chip-select input SCS of the SPI interface, active low level, built-in pull-up resistor. D4: The 4th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
60	SCK/D5	I/O,PU	SCK: Serial clock input SCK of SPI interface, built-in pull-up resistor. D5: The 5th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
61	SDI/D6	I/O,PU	SDI: Serial data input SDI of SPI interface, built-in pull-up resistor. D6: The 6th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
62	SDO/D7	I/O,PU	SDO: Serial data output as SPI interface. D7: The 7th bit in the 8-bit bidirectional data bus with a parallel port, built-in pull-up resistor.
48	ELINK#	O	Network connection indicator LED output: Low level indicates that Ethernet PHY is connected; High level indicates that Ethernet PHY is not connected.
52	EACT#	O	Carrier induction indicates LED output: LED flashing indicates carrier induction signal.

Note 1: Table abbreviation explanation:

I = Input; O = Output; I/O = Input/Output;

P = Power supply; PU = Built-in pull-up resistor.

Note 2: The specific function of the CH395A's alternate pins depends on the currently selected interface mode. When the alternate pin is used for the interface function, its GPIO function will be occupied, resulting in its GPIO function being invalid and the corresponding bit in the GPIO register having no practical meaning.

4.3 CH395P Pin Definition

Table 4-3 CH395P pin definition

CH395P Pin No.	Pin Name	Type(1)	Pin Description
10	RXP	I/O	Differential input in 10BASE-T/100BASE-TX MDI mode; Differential output in 10BASE-T/100BASE-TX MDIX mode.
11	RXN		
14	TXP	I/O	Differential output in 10BASE-T/100BASE-TX MDI mode; Differential input in 10BASE-T/100BASE-TX MDIX mode.
15	TXN		
7/21	VCC33	P	3.3V power supply input, it is recommended to place 0.1uF parallel 10uF or 4.7uF ground capacitor close to the core.
39/56/70/83/101/122	VCCIO	P	For the power input of I/O interface, it is recommended to place 0.1uF capacitance to ground close to the chip.
28	VDDK	P	The external 1uF ground capacitor is placed close to the chip.
54/82	VDDK	P	The external 0.1uF ground capacitor is placed close to the chip.
2/9/16/17/22/27/29/55/64/71/109	GND	P	Common ground.
1/3/4/5/6/8/12/13/18/23/24/25/26/30/31/32/33/34/35/36/37/38/40/41/42/43/44/45/46/47/57/58/65/66/67/68/69/72/73/74/75/76/77/78/79/80/81/84/85/86/87/88/89/90/91/92/93/95/96/97/102/103/105/108/110/111/127/128	NC	-	Reserved, suggested suspension.
19	XI	I	The crystal oscillator input needs an external 25MHz crystal end or an external clock input with a built-in crystal oscillator matching capacitor.
20	XO	O	The inverted output of the crystal oscillator needs to be externally connected to the other end of the 25MHz crystal, and a crystal oscillator matching capacitor is built in.
48	GPIO0	I/O	General-purpose input and output pin 0, configured as input by default.
49	GPIO1	I/O	General-purpose input and output pin 1, configured as input by default.
50	GPIO2	I/O	General-purpose input and output pin 2, configured as input by default.

51	GPIO3/RDY#	I/O	GPIO3: General-purpose input and output pin 3, configured as input by default. RDY#: After CH395P is reset, the output is low.
52	RST	O	Power-on reset and external reset output, active high level.
53	SEL	I,PU	Used as an interface configuration input during internal chip reset. For details on the specific configuration method, please refer to Chapter 6.1.
59	GPIO4	I/O	General-purpose input and output pin 4, configured as input by default.
60	GPIO5	I/O	General-purpose input and output pin 5, configured as input by default.
61	GPIO6	I/O	General-purpose input and output pin 6, configured as input by default.
62	GPIO7	I/O	General-purpose input and output pin 7, configured as input by default.
63	RSTI	I,PU	External reset input, active low, built-in pull-up resistor.
98	INT#	O	Interrupt request output, active low.
99	A0	I,PU	Address input of parallel port, distinguishing command port from data port, built-in pull-up resistor; When A0=1, the command can be written, and when A0=0, the data can be read and written.
100	PCS#	I,PU	Chip selection control input of parallel port, active low, built-in pull-up resistor.
106	RD#	I,PU	Read strobe input of parallel port, active low, built-in pull-up resistor.
107	WR#	I,PU	Write strobe input of parallel port, active low, built-in pull-up resistor.
112	RXD	I,PU	Serial data input of UART with built-in pull-up resistor.
113	TXD	I/O	During the internal reset of the chip, the input is configured for the interface, and the pull-up resistor is built in. For the configuration method, please refer to Chapter 6.1. After the chip reset is completed, it is the serial data output of asynchronous serial port.
114	SCS	I,PU	Chip selection input of SPI interface, active low, built-in pull-up resistor.
115	SCK	I,PU	Serial clock input of SPI interface, built-in pull-up resistor.
116	SDI	I,PU	Serial data input of SPI interface, built-in pull-up resistor.
117	SDO	O	Serial data output of SPI interface.
118~121 123~126	D0 ~ D7	I/O,PU	Parallel port 8-bit bidirectional data bus, built-in pull-up resistor.
94	ELINK#	O	Network connection indicator LED output: Low level indicates that Ethernet PHY is connected; High level indicates that Ethernet PHY is not connected.
104	EACT#	O	Carrier induction indicates LED output: LED flashing indicates carrier induction signal.

Note 1: Table abbreviation explanation:

I = Input; O = Output; I/O = Input/Output;

P = Power supply; PU = Built-in pull-up resistor.

5. Command

The data in this datasheet, the suffix B or b is a binary number, and the suffix H is a hexadecimal number, otherwise it is a decimal number.

The double-word data with the lower byte in front (Little-Endian) (32 bits in total) refers to: first the lowest byte (bits 7 to 0), then the lower byte (bits 15 to 8), then the higher byte (bits 23 to 16), and finally the highest byte (bits 31 to 24).

A data stream refers to a data block composed of several consecutive bytes, with a minimum total length of 0.

The numbers in the brackets of the input data and output data in the following table are the number of bytes of the parameter, and if there is no bracket, the default is 1 byte.

The MCU referred to in this datasheet is basically suitable for DSP or MCU/MPU/SCM, etc.

Socket Pair contains quadruples of source IP, source port, destination IP and destination port, which can uniquely determine the 2 connections in the Internet. This manual is referred to as Socket. CH395 can provide 8 Sockets at the same time, and their index values are 0, 1, 2, 3, 4, 5, 6, and 7 respectively.

The IP and MAC addresses agreed upon in this manual may differ from some documents for convenience only:

If the IP address is 192.168.1.2, where 192 is the lowest byte and 2 is the highest byte. This article says (IP) low bytes are in front.

If the MAC address is 00.01.02.03.04.05, 00 is the lowest byte and 05 is the highest byte. This paper calls (MAC) low byte first.

In this datasheet, all commands containing IP address input or output are in byte order with IP low byte first.

In this datasheet, all commands containing MAC address input or output are in byte order with MAC low byte first.

Table 5-1 Command list

Code	Command name CMD_	Input data	Output data	Command purpose
01H	GET_IC_VER		Version number	Get the chip and firmware version
02H	SET_BAUDRATE	Baud rate coefficient (3)	Operating state	Set the serial port communication baud rate
03H	ENTER_SLEEP			Enter low-power sleep suspend state
05H	RESET_ALL		(Wait 15ms)	Perform a hardware reset
06H	CHECK_EXIST	Any data	Bitwise invert	Test communication interface and working status
19H	GET_GLOB_INT_STATUS _ALL		Global interrupt status (2)	Get global interrupt status
20H	SET_PHY	PHY connection method		Set the PHY connection method
21H	SET_MAC_ADDR	MAC address (6)		Set MAC address
22H	SET_IP_ADDR	IP address (4)		Set IP address

23H	SET_GWIP_ADDR	Gateway address (4)		Set the gateway IP address
24H	SET_MASK_ADDR	Subnet mask (4)		Set subnet mask
25H	SET_MAC_FILT	Filter mode HASH0(4) HASH1(4)		Set MAC filtering mode
26H	GET_PHY_STATUS		PHY status	Get PHY status
27H	INIT_CH395			Initialize CH395 chip
28H	GET_UNREACH_IPPORT		Unreachable information (8)	Get unreachable IP, port and protocol
29H	GET_GLOB_INT_STATUS		Global interrupt status	Get global interrupt status
2AH	SET_RETRAN_COUNT	Retry times		Set retry number, up to 20 times
2BH	SET_RETRAN_PERIOD	Retry cycle (2)		Set retry period, up to 1000ms
2CH	GET_CMD_STATUS		Command execution status	Get command execution status
2DH	GET_REMOT_IPP_SN	Socket index	IP and port (6)	Get the IP and port of the remote end (destination)
2EH	CLEAR_RECV_BUF_SN	Socket index		Clear the receive buffer of Socket
2FH	GET_SOCKET_STATUS_SN	Socket index	Socket status	Get Socket status
30H	GET_INT_STATUS_SN	Socket index	Socket interrupt	Get Socket interrupt status
31H	SET_IP_ADDR_SN	Socket index Destination IP (4)		Set destination IP address of Socket.
32H	SET_DES_PORT_SN	Socket index Destination port (2)		Set destination port address of Socket.
33H	SET_SOUR_PORT_SN	Socket index Source port (2)		Set source port of Socket
34H	SET_PROTO_TYPE_SN	Socket index Protocol type		Set operation mode of Socket
35H	OPEN_SOCKET_SN	Socket index		Open Socket
36H	TCP_LISTEN_SN	Socket index		Enable Socket listening
37H	TCP_CONNECT_SN	Socket index		Enable Socket connection
38H	TCP_DISCONNECT_SN	Socket index		Disconnect the TCP connection of Socket
39H	WRITE_SEND_BUF_SN	Socket index Length (2) Data flow (N)		Send buffer write data to Socket
3BH	GET_RECV_LEN_SN	Socket index	Length (2)	Gets Socket received data length
3CH	READ_RECV_BUF_SN	Socket index Length (2)	Data flow (N)	Receive data from Socket receive buffer
3DH	CLOSE_SOCKET_SN	Socket index		Disable Socket
3EH	SET_IPRAW_PRO_SN	Socket index IP protocol field		Set the protocol field of the IP packet of Socket.

3FH	PING_ENABLE	Enable flag		PING enable
40H	GET_MAC_ADDR		MAC address (6)	Get MAC address
41H	DHCP_ENABLE	Enable flag		Start (Stop) DHCP
42H	GET_DHCP_STATUS		DHCP status	Get DHCP status
43H	GET_IP_INF		IP and other information	Get IP, MASK, DNS and other information.
44H	SET_ARP	ARP retransmission period ARP retransmission number		Set the ARP retransmission cycle and times.
50H	SET_TCP_MSS	TCP MSS (2)		Set TCP MSS
51H	SET_TTL	TTL		Set TTL value, up to 128.
52H	SET_RECV_BUF	Socket index Start block address Block number		Set Socket receive buffer
53H	SET_SEND_BUF	Socket index Start block address Block number		Set Socket send buffer
55H	SET_FUN_PARA	Function parameter (4)		Set functional parameter
56H	SET_KEEPALIVE_IDLE	4-byte time parameter		Set KEEPALIVE idle time
57H	SET_KEEPALIVE_INTVL	4-byte time parameter		Set KEEPALIVE timeout
58H	SET_KEEPALIVE_CNT	Retry times		Set the number of KEEPALIVE timeout retries
59H	SET_KEEPALIVE_SN	Socket index Configuration		Set Socket KEEPALIVE
E9H	EEPROM_ERASE			Erase EEPROM
EAH	EEPROM_WRITE	EEPROM address (2) Length Data flow (N)		Write EEPROM
EBH	EEPROM_READ	EEPROM address (2) Length	Data flow (N)	Read EEPROM
ECH	READ_GPIO_REG	GPIP0 register address	GPI0 register value	Read GPIO register
EDH	WRITE_GPIO_REG	GPIP0 register address GPIP0 register value		Write GPIO register

Commands in the shaded part of the table usually need to be executed for a certain period of time and the execution status of the command is queried. The MCU can obtain the status through GET_CMD_STATUS (refer to CH395INC.H for the definition of status).

5.1 CMD_GET_IC_VER

This command is used to obtain the chip and firmware version, and the returned 1 byte data is the version number.

5.2 CMD_SET_BAUDRATE

This command is used to set the baud rate of serial communication of CH395. When CH395 works in serial communication mode, the default communication baud rate after reset is set by the level combination of SDO,

SDI and SCK pins (refer to Section 6.3 of this datasheet), and the default communication baud rate is 9600bps when these pins are suspended. If the MCU supports high communication speed, the baud rate of serial communication can be dynamically adjusted by this command. This command needs to input three data, namely baud rate coefficient 0, baud rate coefficient 1 and baud rate coefficient 2. The following table shows the corresponding relationship with baud rate.

Table 5-2 Baud rate

Baud rate coefficient 2	Baud rate coefficient 1	Baud rate coefficient 0	Serial port communication baud rate (bps)
00H	12H	C0H	4800
00H	25H	80H	9600
00H	4BH	00H	19200
00H	96H	00H	38400
00H	E1H	00H	57600
01H	2CH	00H	76800
01H	C2H	00H	115200
07H	08H	00H	460800
0EH	10H	00H	921600
0FH	42H	40H	1M
1EH	84H	80H	2M
4CH	4BH	40H	5M
98H	96H	80H	10M
Calculation formula: BaudRate = (Baud rate coefficient 2<<16) + (Baud rate coefficient 1<<8) + Baud rate coefficient 0			

Set the execution time of the serial communication baud rate configuration command by default to TE8 (the minimum is 1ms, see Table 7-4). After completion, CH395 will output the operating state at the new baud rate. The MCU needs to adjust its own baud rate as soon as possible within TE8 time and wait for CH395 to answer.

5.3 CMD_ENTER_SLEEP

This command causes the CH395 chip to enter a low-power sleep suspend state. Pull the CH395RXD pin down by 35us or above in serial port mode, and pull the SCS or PCS# chip select pin down by 35us or above in SPI or parallel port mode to remove the CH395 from the low power state. Therefore, after the MCU issues the CMD_ENTER_SLEEP command, the SCS or PCS# chip select should be invalid immediately.

The MAC and PHY that enter sleep state CH395 will enter power-down mode and disconnect the Ethernet connection.

After waking up from sleep mode, the CH395 takes about 4ms to complete the wake-up process. During this wake-up phase, CH395 will not parse or respond to any received command code. Therefore, do not send any command code to CH395 during CH395 being awakened. It is recommended that the user wake up for 4ms before operating on CH395.

5.4 CMD_RESET_ALL

This command causes CH395 to perform a hardware reset. Normally, hardware reset is completed within 15ms.

5.5 CMD_CHECK_EXIST

This command is used to test the communication interface and working status to check whether the CH395 is working properly. This command requires input of 1 byte of data, which can be any data. If CH395 works normally, then the output data of CH395 is bitwise inverse of the input data. For example, if the input data is 57H, the output data is A8H.

5.6 CMD_SET_PHY

This command uses the connection method to set the CH395 Ethernet PHY. The default is automatic negotiation method. This command requires inputting 1 byte of data, which is the connection method code:
Disconnect PHY when connection mode code is 01H;
PHY is 10M full duplex when the connection mode code is 02H;
PHY is 10M half-duplex when the connection mode code is 04H;
PHY is 100M full-duplex When the connection mode code is 08H;
PHY is 100M half-duplex when the connection mode code is 10H;
PHY auto-negotiation when the connection mode code is 20H.
After receiving this command, CH395 will reset the MAC and PHY and reconnect according to the newly set connection mode. If Ethernet is connected, it will be disconnected and reconnected.

5.7 CMD_SET_MAC_ADDR

This command is used to set the MAC address of CH395. You need to enter a 6-byte MAC, and the low byte of the MAC address comes first. The CH395 chip will store the MAC address in the internal EEPROM. This command takes about 30us to complete.

The CH395 chip has been programmed with the MAC address assigned by IEEE when it leaves the factory. Please do not set the MAC address unless it is necessary.

5.8 CMD_SET_IP_ADDR

This command is used to set the IP address of CH395. You need to enter a 4-byte IP address, with the IP low byte first. In this manual, all commands containing IP input or output are in the byte order of IP low byte first, which will not be explained below.

5.9 CMD_SET_GWIP_ADDR

This command is used to set the gateway address of CH395. You need to enter a 4-byte IP address.

5.10 CMD_SET_MASK_ADDR

This command is used to set the subnet mask of CH395. This command requires a 4-byte mask. The default value is 255.255.255.0, which can be left unchecked.

5.11 CMD_SET_MAC_FILT

This command is used to set the MAC filtering mode. The MAC can set multiple filtering modes. This command needs to input 9 bytes of data, the first byte is filter mode, and the meaning of each bit of this data is as follows:

Table 5-3 Baud rate

Bit	Name	Description
[5:7]	-	Reserved

4	SEND_ENABLE	Send enable
3	RECV_ENABLE	Receive enable
2	RECV_MULTIPKT	Receive multicast packets
1	RECV_ALL	Receive all data
0	RECV_BROADPKT	Receive broadcast packet

The above bits are 1 for enable, and 0 for disable. After CH395 is reset, RECV_BROADPKT, RECV_ENABLE and SEND_ENABLE are enabled by default.

The following table shows what you mean:

Table 5-4 MAC filtering meaning

RECV_ENABLE	RECV_ALL	RECV_BROADPKT	RECV_MULTIPKT	Description
1	0	0	0	Receive packets matching the MAC address
1	1	X	X	Receive all packets
1	0	1	0	Receive packets matching the MAC address Receive broadcast packets
1	0	0	1	Receive packets matching the MAC address Receive multicast packets
1	0	1	1	Receive packets matching the MAC address Receive broadcast packets Receive multicast packets
0	X	X	X	Prohibit receiving

The 2nd to 5th bytes are HASH0 (Hash Table 0), and the 6th to 9th bytes are Hash H1 (Hash Table 1). HASH0 and Hash H1 are only valid when multicast is turned on.

HASH0 and HASH1 together form a 64-bit HASH table, with 0-31 bits being HASH0 and 32-63 bits being HASH1.

HASH table calculation method: calculate a 32-bit CRC value for the multicast address by using standard Ethernet redundancy check (CRC32), use the upper 6 bits of this CRC value as the index value, and write 1 to the corresponding bit of the HASH table. For example, if the upper 6 bits of the CRC value calculated by the multicast address are 32, the 0 th bit of HASH1 should be written as 1.

5.12 CMD_GET_PHY_STATUS

This command is used to obtain the connection status of PHY. After receiving this command, CH395 will query the current connection status of PHY and output a 1-byte PHY connection status code:
A connection status code of 01H indicates that the PHY connection is disconnected;
A connection status code of 02H indicates that the PHY connection is 10M full-duplex;
A connection status code of 04H indicates that the PHY connection is 10M half-duplex;
A connection status code of 08H indicates that the PHY connection is 100M full-duplex;
A connection status code of 10H indicates that the PHY connection is 100M half-duplex.

5.13 CMD_INIT_CH395

This command is used to initialize CH395, including initializing the MAC, PHY and TCP/IP protocol stack of CH395. This command takes about 5ms to complete. MCU can send GET_CMD_STATUS to inquire whether the execution is completed and the execution status.

5.14 CMD_GET_UNREACH_IPPORT

This command is used to obtain unreachable IP, port and protocol type. When an unreachable message is received, CH395 will generate an unreachable interrupt. MCU can use this command to obtain unreachable information. After receiving this command, CH395 will output 1-byte unreachable code, 1 byte protocol type, 2 bytes port number (low byte first) and 4 bytes IP in turn. MCU can judge whether the protocol is unreachable, the port is unreachable or the IP is unreachable according to the unreachable code. For unreachable codes, please refer to RFC792 (CH395INC.H defines four common unreachable codes).

5.15 CMD_GET_GLOB_INT_STATUS

This command is used to obtain the global interrupt status. After receiving this command, CH395 will output the global interrupt status of 1-byte. The global interrupt status is defined as follows:

Table 5-5 Global interrupt 1

Bit	Name	Description
7	GINT_STAT_SOCK3	Socket3 interrupt
6	GINT_STAT_SOCK2	Socket2 interrupt
5	GINT_STAT_SOCK1	Socket1 interrupt
4	GINT_STAT_SOCK0	Socket0 interrupt
3	GINT_STAT_DHCP	DHCP interrupt
2	GINT_STAT_PHY_CHANGE	PHY state change interrupt
1	GINT_STAT_IP_CONFLI	IP conflict
0	GINT_STAT_UNREACH	Unreachable interrupt

① GINT_STAT_UNREACH: unreachable interrupt. When CH395 receives the ICMP unreachable interrupt message, it saves the IP address, port and protocol type of the unreachable IP packet in the unreachable information table, and then generates this interrupt. When the MCU receives this interrupt, it can send the GET_UNREACH_IPPORT command to obtain the unreachable information.

② Gint_stat_IP_conf Li: IP conflict interrupt. This interrupt occurs when CH395 detects that its own IP address is the same as that of other network devices in the same network segment.

③ Gint_stat_phy_change: PHY change is interrupted. This interrupt occurs when the PHY connection of CH395 changes, for example, the PHY state changes from the connected state to the disconnected state or from the disconnected state to the connected state. The MCU can send the GET_PHY_STATUS command to get the status of the current PHY connection.

④ Gint_stat_DHCP: DHCP interrupt. If the MCU enables the DHCP function of CH395, CH395 will generate this interrupt after DHCP succeeds or times out, and the MCU can send CMD_GET_DHCP_STATUS command to obtain the DHCP status. If the status is 0, it means success, otherwise it means timeout failure.

⑤ Gint_stat_sock 0 ~ gint_stat_sock 3: socket interrupt. CH395 will generate this interrupt when there is an interrupt event in the Socket, and the MCU needs to send GET_INT_STATUS_SN to get the interrupt status of the Socket. Please refer to the GET_INT_STATUS_SN section.

After this command is executed, CH395 will set the INT# pin high and clear the global interrupt status.

5.16 CMD_GET_GLOB_INT_STATUS_ALL

This command is used to obtain the global interrupt status. After receiving this command, CH395 will output the global interrupt status of 2 bytes. The global interrupt status is defined as follows:

Table 5-6 Global interrupt 2

Bit	Name	Description
[12:15]	-	Reserved
11	GINT_STAT_SOCK7	Socket7 interrupt
10	GINT_STAT_SOCK6	Socket6 interrupt
9	GINT_STAT_SOCK5	Socket5 interrupt
8	GINT_STAT_SOCK4	Socket4 interrupt
7	GINT_STAT_SOCK3	Socket3 interrupt
6	GINT_STAT_SOCK2	Socket2 interrupt
5	GINT_STAT_SOCK1	Socket1 interrupt
4	GINT_STAT_SOCK0	Socket0 interrupt
3	GINT_STAT_DHCP	DHCP interrupt
2	GINT_STAT_PHY_CHANGE	PHY state change interrupt
1	GINT_STAT_IP_CONFLI	IP conflict
0	GINT_STAT_UNREACH	Unreachable interrupt

For bits 0-7, please refer to section 5.15.

Gint_stat_sock 4 ~ gint_stat_sock 7: socket interrupt. CH395 will generate this interrupt when there is an interrupt event in the Socket, and the MCU needs to send GET_INT_STATUS_SN to get the interrupt status of the Socket. Please refer to the GET_INT_STATUS_SN section.

CH395 can obtain the interrupt status by 2 commands, CMD_GET_GLOB_INT_STATUS and CMD_GET_GLOB_INT_STATUS_ALL. The former can only obtain the interrupt status of the lower 8 bits, and the latter can obtain all the interrupt status. Please pay attention to it when using it. Any version of the chip supports the command CMD_GET_GLOB_INT_STATUS. If the chip version number is greater than or equal to 0X44 and Socket4~Socke7 is used, only CMD_GET_GLOB_INT_STATUS_ALL can be used. CMD_GET_GLOB_INT_STATUS_ALL command is not supported if the chip version number is less than 0X44.

5.17 CMD_SET_RETRAN_COUNT

This command is used to set the number of retries. You need to enter the number of retries of 1 byte, and the maximum allowable value is 20. If the input data is greater than 20, it will be treated as 20. The default number of retries is 12, and retries are only valid in TCP mode.

5.18 CMD_SET_RETRAN_PERIOD

This command is used to set the retry cycle. You need to enter the cycle number of 2 bytes (low byte first) in ms, and the maximum allowable value is 1000. The total time of retry is N*M, where n is the number of retries and m is the retry period. The default retry period is 500ms, and retry is only valid in TCP mode.

5.19 CMD_SET_RETRAN_PERIOD

This command is used to obtain the status of command execution. CH395 will output 1 byte of data, which is the state of command execution. Command execution status is as follows:

Table 5-7 Command execution status

Code	Name	Description
00H	CH395_ERR_SUCCESS	Success
10H	CH395_ERR_BUSY	Busy, indicating that the command is being executed
11H	CH395_ERR_MEM	Memory management error
12H	CH395_ERR_BUF	Buffer error
13H	CH395_ERR_TIMEOUT	Timeout
14H	CH395_ERR_RTE	Routing error
15H	CH395_ERR_ABRT	Connection aborted
16H	CH395_ERR_RST	Connection reset
17H	CH395_ERR_CLSD	Connection closed
18H	CH395_ERR_CONN	No connection
19H	CH395_ERR_VAL	Error value
1AH	CH395_ERR_ARG	Error parameters
1BH	CH395_ERR_USE	Has been used
1CH	CH395_ERR_IF	MAC error
1DH	CH395_ERR_ISCONN	Connected
20H	CH395_ERR_OPEN	Opened

If the MCU receives CH395_ERR_BUSY, it means that CH395 is executing the command, and the MCU should delay for more than 2ms and send the CMD_GET_CMD_STATUS command again to get the status.

Commands in the shaded part of the command code table need to send CMD_GET_CMD_STATUS to get the execution status.

5.20 CMD_GET_REMOT_IPP_SN

This command is used to obtain the IP address and port number of the remote end. It needs to input a 1-byte Socket index value, and CH395 will output 4-byte IP address and 2-byte (low byte first) port number. After the Socket works in TCP Server mode and the connection is established, the MCU can obtain the remote IP address and port number through this command.

5.21 CMD_CLEAR_RECV_BUF_SN

This command is used to empty the receive buffer of Socket, and a 1-byte Socket index value is required. Upon receiving this command, CH395 will clear the receiving length of this Socket, and the receiving pointer will point to the head of the buffer.

5.22 CMD_GET_SOCKET_STATUS_SN

This command is used to obtain the status of Socket, and it needs to input a Socket index value of 1 byte. CH395 will output a 2-byte status code after receiving this command.

The first status code is the status code of Socket, and the status code of Socket is defined as follows:

Table 5-8 Socket status code

Code	Name
00H	SOCKET_CLOSED

05H	SOCKET_OPEN

The second status code is the status code of TCP, which is meaningful only when it is in TCP mode and has been turned on. The TCP status code is defined as follows:

Table 5-9 TCP status code

Code	Name	Description
00H	TCP_CLOSED	Closed
01H	TCP_LISTEN	Listen
02H	TCP_SYN_SENT	SYN send
03H	TCP_SYN_RCVD	SYN receive
04H	TCP_ESTABLISHED	TCP connection establishment
05H	TCP_FIN_WAIT_1	Active shutdown party sends FIN for the first time
06H	TCP_FIN_WAIT_2	Active shutdown party receives ACK from FIN
07H	TCP_CLOSE_WAIT	Passive closing party receives FIN
08H	TCP_CLOSING	Closing
09H	TCP_LAST_ACK	Passive shutdown party sends FIN
0AH	TCP_TIME_WAIT	2MLS waiting state

TCP states are all specified in the TCP/IP protocol. Please refer to the TCP/IP protocol for details.
Only care about 'TCP_CLOSED', 'TCP_LISTEN' and 'TCP_ESTABLISHED' in application. Other status chips will be automatically processed and updated.

5.23 CMD_GET_INT_STATUS_SN

This command is used to obtain the interrupt status of the Socket, and it is required to input a Socket index value of 1 byte. After receiving this command, CH395 will output an interrupt code of 1 byte of the Socket, and each bit of the interrupt code is defined as follows:

Table 5-10 Socket interrupt status

Bit	Name	Description
7	-	Reserved
6	SINT_STAT_TIM_OUT	Timeout
5	-	Reserved
4	SINT_STAT_DISCONNECT	TCP disconnect
3	SINT_STAT_CONNECT	TCP connect
2	SINT_STAT_RECV	Receive buffer is not empty
1	SINT_STAT_SEND_OK	Send success
0	SINT_STAT_SENBUF_FREE	Send buffer free

① SINT_STAT_SENBUF_FREE, the sending buffer is idle. After the MCU writes data to the Socket sending buffer, CH395 will quickly copy the data to the internal protocol stack or MAC buffer to encapsulate the data. When the data is copied, this interrupt will occur, and the MCU can continue to write subsequent data to the sending buffer. After the MCU writes data to the Socket sending buffer once, it must wait until this interrupt occurs before writing the next data.

② SINT_STAT_SEND_OK, interrupt of successful transmission. This interrupt is generated, indicating that the data packet was successfully transmitted. Sokcet will generate this interrupt every time a packet of data is

successfully transmitted. After the MCU writes data to the Socket buffer once, CH395 may be packaged into several data packets for transmission, so there may be several successful transmission interrupts.

③ SINT_STAT_CONNECT, TCP connection is broken, only valid in TCP mode. It shows that the TCP connection is successful, and the MCU must interrupt before data transmission can be carried out.

④ SINT_STAT_DISCONNECT, TCP connection disconnection interrupt, valid only in TCP mode, indicating that TCP connection is disconnected.

⑤ SINT_STAT_TIM_OUT, in TCP mode, this interrupt will be generated when timeout occurs during TCP connection, disconnection, data transmission, etc. In IPRAW and UDP modes, this interrupt will also be generated when sending data fails.

After the interrupt of SINT_STAT_DISCONNECT and SINT_STAT_TIM_OUT is generated, CH395 will make different actions according to whether the FUN_PARA_FLAG_SOCKET_CLOSE bit is 1 or 0. If FUN_PARA_FLAG_SOCKET_CLOSE is 0, when the above two interrupts are generated, CH395 will actively set the Socket state to the closed state and clear all related buffers. On the other hand, it will not do anything about the Socket status and related buffers, so as to facilitate the external MCU to read the residual data after TCP disconnection or timeout. When the external MCU reads the data, it must send a close command to close the Socket.

5.24 CMD_SET_IP_ADDR_SN

This command is used to set the Socket destination IP address. You need to enter a Socket index value of 1 byte and a destination IP address of 4 bytes. When the Socket works in IPRAW, UDP and TCP Client modes, the destination IP must be set before sending the CMD_OPEN_SOCKET_SN command.

5.25 CMD_SET_DES_PORT_SN

This command is used to set the Socket destination port. You need to enter the Socket index value of 1 byte and the destination port of 2 bytes (the lower byte comes first). Socket works in UDP and TCPClient mode, and this value must be set.

5.26 CMD_SET_SOUR_PORT_SN

This command is used to set the Socket source port. You need to enter a Socket index value of 1 byte and a source port of 2 bytes (low byte first). If two or more Socket use the same mode, the source port numbers must not be the same. For example, Socket0 is in UDP mode, the source port is 600, and Socket1 is also in UDP mode. You can't use the source port 600 again, otherwise it may lead to opening failure.

5.27 CMD_SET_PROTO_TYPE_SN

This command is used to set the working mode of Socket, and it is required to input a Socket index value of 1 byte and a working mode of 1 byte. The working mode is defined as follows:

Table 5-11 Socket working mode

Code	Name	Description
03H	PROTO_TYPE_TCP	TCP mode
02H	PROTO_TYPE_UDP	UDP mode
01H	PROTO_TYPE_MAC_RAW	MAC original message mode
00H	PROTO_TYPE_IP_RAW	IP original message mode

This command must be executed before CMD_OPEN_SOCKET_SN. For detailed steps, please refer to 9.2 Application Reference Steps.

5.28 CMD_OPEN_SOCKET_SN

This command is used to open the Socket, which is a necessary step to use the Socket. You need to enter a Socket index value of 1 byte. After sending this command, the MCU should send GET_CMD_STATUS to query the execution status of the command. After opening the Socket in UDP, IPRAW and MACRAW modes and returning successfully, data transmission can be carried out. Before sending this command, necessary settings must be made for the destination IP, protocol type, source port and destination port. For detailed steps, please refer to 9.2 Application Reference Steps.

5.29 CMD_TCP_LISTEN_SN

This command is only valid in TCP mode, enabling Socket to enter the listening mode, that is, TCP Server mode. You need to enter a 1-byte Socket index, and this command must be executed after OPEN_SOCKET_SN. After sending this command, the MCU should send GET_CMD_STATUS to inquire about the execution status of the command.

Under the TCP Server mode, the Socket will directly test the connection event until the connection is completed, and the SINT_STAT_CONNECT interrupt will be generated. Only one connection can be established per Socket. If a qualified connection event is received again, Socket will send TCP RESET to the remote end trying to connect.

5.30 CMD_TCP_CONNECT_SN

This command is only valid in TCP mode. To enable Socket to enter the connection mode, that is, TCP Client mode, you need to enter a Socket index value of 1 byte. After sending this command, the MCU should send GET_CMD_STATUS to inquire about the execution status of the command.

After receiving this command, Socket will initiate a connection event, and after successful connection, it will generate a SINT_STAT_CONNECT interrupt. SINT_STAT_TIM_OUT interrupt will be generated if there is an exception in the connection process or if the connection fails after a certain period of time. The MCU receives this interrupt, and if it needs to be connected again, it needs to reopen the Socket and perform TCP_CONNECT_SN.

5.31 CMD_TCP_DISCONNECT_SN

This command is only valid in TCP mode. To disconnect the current TCP connection, you need to enter a 1-byte Socket index value. After sending this command, the MCU should send GET_CMD_STATUS to inquire about the execution status of the command. SINT_STAT_DISCONNECT interrupt will be generated after the current TCP is successfully disconnected.

5.32 CMD_WRITE_SEND_BUF_SN

This command is used to write data to the send buffer of Socket. It needs to input the Socket index value of 1 byte, the length of 2 bytes (low byte first) and several bytes of data stream. The length of the input data must not be greater than the size of the send buffer, but in MACRAW mode, the maximum length of the input data can only be 1514, and the redundant data will be discarded. When the external MCU finishes writing the data, CH395 will package the data packet according to the working mode of Socket, and then send it. Before the MCU receives SINT_STAT_SENBUF_FREE, it is not allowed to write data to the Socket sending buffer again.

5.33 CMD_GET_RECV_LEN_SN

This command is used to obtain the effective data length of the current receiving buffer. It needs to input a Socket index value of 1 byte. After receiving this command, CH395 outputs a length of 2 bytes (the low byte comes first).

5.34 CMD_READ_RECV_BUF_SN

This command is used to read data from the Socket receiving buffer. It needs to input a Socket index value of 1 byte, and the length of 2 bytes (the lower byte comes first). CH395 will output several bytes of data stream according to the length value. In practical application, you can send the RECV_LEN_SN command first to obtain the actual effective length of the current buffer. The length of the read data can be less than the actual effective data length of the buffer, and the unread data still remains in the receiving buffer, and the MCU can continue reading through this command.

In MACRAW mode, the processing methods are different. In MACRAW mode, the receiving buffer is a frame buffer, and only one frame of Ethernet data can be cached. After processing the command READ_RECV_BUF_SN, CH395 will clear all the receiving buffers of Socket0, so the MCU should read all the valid data of the buffer at one time.

5.35 CMD_CLOSE_SOCKET_SN

This command is used to close the Socket. You need to enter a Socket index value of 1 byte. After closing the Socket, the receiving and sending buffer of the Socket is emptied, but the configuration information is still retained. You only need to open the Socket again the next time you use the Socket.

In TCP mode, CH395 will automatically disconnect the TCP connection before closing the Socket.

5.36 CMD_SET_IPRAW_PRO_SN

This command is valid only in IPRAW mode and requires the entry of a 1-byte socket index value, and a 1-byte protocol code for the IP Packet Protocol field. This command must be executed before OPEN_SOCKET_SN.

If multiple Socket adopt IPRAW mode, the protocol code shall not be reused. For example, if Socket 0 and Socket 1 both adopt IPRAW mode, the protocol codes of the two sockets shall not be the same, otherwise it may lead to failure when opening the socket.

IPRAW data processing priority is higher than UDP and TCP, so the protocol code must not be the same as other Socket. For example, Socket 0 adopts IPRAW mode and the protocol code is 17(UDP protocol). Socket 1 adopts UDP mode. This may cause the data of Socket 1 to be received by Socket 0.

5.37 CMD_PING_ENABLE

This command is used to turn PING on or off, and you need to enter a 1-byte flag code. A flag code of 1 means to turn PING on, and a flag code of 0 means to turn PING off.

5.38 CMD_GET_MAC_ADDR

This command is used to obtain the MAC address. After receiving this command, CH395 will output a 6-byte MAC address.

5.39 CMD_DHCP_ENABLE

This command is used to start or stop DHCP. You need to enter a 1-byte flag code. If the flag code is 1, it means to start DHCP. If it is 0, it means to close DHCP. Before starting DHCP, you must initialize CH395.

After starting DHCP, CH395 will broadcast DHCPDISCOVER message to the network, which is used to discover DHCP Server, apply for address and other configuration parameters after finding DHCP Server, and then generate GINT_STAT_DHCP interrupt. After this interrupt is generated, the MCU can send the command GET_DHCP_STATUS to get the status of DHCP. If the status is 0, it means success, then the MCU can send the command GET_IP_INF to obtain information such as IP and MASK. If the status is 1, it indicates an error, which is usually caused by timeout, for example, DHCP Server is not found.

DHCP is always in working state after it is started, unless it receives a command from the MCU to turn off DHCP. During this process, if DHCP Server reassigns a configuration to CH395, and this configuration is different from the original one, CH395 will still generate an interrupt.

After the timeout interrupt occurs, if the DHCP Server is not found, CH395 will continue to send DHCPDISCOVER messages with an interval of about 16 seconds.

This command takes about 1ms to execute, and the MCU can send GET_CMD_STATUS to inquire whether it has been executed and its execution status.

5.40 CMD_GET_DHCP_STATUS

This command is used to obtain the status of DHCP. Generally, after the MCU receives the interrupt, it sends this command to obtain the execution status of DHCP. After receiving this command, CH395 outputs the DHCP status code. There are two status codes, namely 0 and 1, which are as follows:

If the status is 0, it means success, then the MCU can send the command GET_IP_INF to obtain information such as IP and MASK.

If the status is 1, it indicates an error, which is usually caused by timeout, for example, DHCP Server is not found.

5.41 CMD_GET_IP_INF

This command is used to obtain information such as IP, GatewayIP, MASK, DNS, etc. After receiving this command, CH395 will output 20 bytes of data in turn, namely: 4 bytes of IP address, 4 bytes of gateway IP, 4 bytes of subnet mask, 4 bytes of DNS1 (primary DNS) and 4 bytes of DNS2 (secondary DNS).

After DHCP, you can send this command to get the information of the current CH395. If some configurations are not obtained in DHCP, the configuration is 0. For example, DNS may not be allocated when DHCP is used in local area network. When this command is sent to obtain configuration information, DNS1 and DNS2 are all 0.

5.42 CMD_SET_ARP

This command is used to set the ARP retransmission cycle and times. You need to enter a 1-byte retransmission cycle n (100ms) and a 1-byte retransmission number m (times). CH395F, CH395A and CH395P support this command.

When CH395 sends data to the target device, it will search the APR cache table to find the MAC address of the target device. If it is not stored in the cache table, CH395 will send an ARP request to obtain the MAC address of

the target device. If the target device does not respond, CH395 will retransmit according to the set retransmission period n (100ms), with the maximum retransmission time of m times.

By default, the number of retransmissions is 3 (times) and the retransmission period is 10 (that is, 1 second). If the ARP retransmission number or period is set to 0, the default value will be used. If the number of retransmissions is set to 255, it means that CH395 will retransmit indefinitely until the MAC address of the target device is successfully obtained.

This command should be set after initializing CH395.

5.43 CMD_SET_TCP_MSS

This command is used to set TCPMSS, and a 2-byte TCPMSS value is required, with a maximum value of 1460 and a minimum value of 60. This value should be set before initializing CH395, and it is not allowed to be set after initialization.

5.44 CMD_SET_TTL

This command is used to set the TTL of Socket, and it needs to input a 1-byte Socket index and a 1-byte TTL value. It should be set after opening the Socket, and the maximum value is 128. This command does not take effect when Socket is in TCP Sever mode.

5.45 CMD_SET_RECV_BUF

This command is used to set the receiving buffer of Socket, and 3 bytes of data are required, the first byte is the Socket index, the second byte is the starting block of the buffer, and the third byte is the number of blocks.

Block 0	Block 1	Block 2	……	Block 46	Block 47

The internal buffer structure of CH395 is shown above, which consists of 48 blocks, each of which is 512 bytes long. MCU can freely allocate the size of each Socket receiving buffer. The allocation of buffer after initialization of CH395 is as follows:

Table 5-12 CH395 internal buffer

Socket	Buffer	Start block	Number of blocks
0	Receive buffer	0	8
	Send buffer	8	4
1	Receive buffer	12	8
	Send buffer	20	4
2	Receive buffer	24	8
	Send buffer	32	4
3	Receive buffer	36	8
	Send buffer	44	4
4	Receive buffer	NULL (Empty, not allocated, the same below)	0
	Send buffer	NULL	0
5	Receive buffer	NULL	0
	Send buffer	NULL	0
6	Receive buffer	NULL	0

	Send buffer	NULL	0
7	Receive buffer	NULL	0
	Send buffer	NULL	0

As can be seen from the above table, after CH395 is initialized, all buffers are allocated to Socket0~3, the receiving buffer is 8 blocks (4KB), and the sending buffer is 4 blocks (2KB). If the number of sockets required by the MCU is greater than 4, the buffer needs to be allocated again.

5.46 CMD_SET_SEND_BUF

This command is used to set the Socket's sending buffer. It requires input of 3 bytes of data. The first byte is the Socket index, the second byte is the starting block of the buffer, and the third byte is the number of blocks.

For the definition and allocation of buffers, please refer to Chapter 5.45.

5.47 CMD_EEPROM_ERASE

This command is used to erase the EEPROM. The CH395 chip comes with 4KB of EEPROM, and all the data after erasing is 0XF5. Before writing to EEPROM, it is necessary to ensure that all data in the destination area is 0XF5. This command needs to be executed approximately 21us.

5.48 CMD_EEPROM_WRITE

This command is used to write EEPROM. It requires inputting 2 bytes of address, 1 byte length and several bytes of data stream. The length of the byte stream must not be greater than 64 bytes. This command needs to be executed approximately 6.5us.

5.49 CMD_EEPROM_READ

This command is used to read EEPROM. It requires inputting the address of 2 bytes and the length of 1 byte. CH395 outputs several bytes of data stream according to the length, and the length value must not be greater than 64 bytes. After sending the length, the external MCU should wait 27us before reading the data. This command needs to be executed approximately 5.2us.

5.50 CMD_READ_GPIO_REG

This command is used to read the GPIO register. It requires inputting a register address of 1 byte, and CH395 outputs a register value of 1 byte. All registers are 8 bits, bit0~7 corresponds to GPIO0~7 respectively. Their addresses and meanings are as follows:

Table 5-13 GPIO register

Address	Name	Description
80H	GPIO_DIR_REG	Direction Register 1: Output; 0: Input
81H	GPIO_IN_REG	Input data register
82H	GPIO_OUT_REG	Output data register
83H	GPIO_CLR_REG	0: Keep; 1: Clear
84H	GPIO_PU_REG	Pull-down register 1: Pull-up enable; 0: Pull-up disable
85H	GPIO_PD_REG	Pull-up register 1: Pull-down enable; 0: Pull-down disable

CH395 has 8 channels of GPIO, and GPIO3 defaults to output. After CH395 is successfully initialized, it outputs low level, and the rest GPIO defaults to input state.

5.51 CMD_WRITE_GPIO_REG

This command is used to write a GPIO register, and it needs to input a register address of 1 byte and a register value of 1 byte.

Refer to Section 5.50 for the definition of GPIO register.

5.52 CMD_SET_FUN_PARA

This command is used to set function parameters, and 4-byte parameters are required. The meanings of each parameter are as follows:

Table 5-14 Functional parameters

Bit	Name	Reset value	Description
[31:8]	-	-	Reserved, must be 0
8	FUN_PARA_FLAG_UART_FLOW_CONTROL	0	Serial port flow control enable bit, only supported by CH395F
7	FUN_PARA_FLAG_UART_FAST_DEAL	0	Serial port fast reply enable bit, supported by 0x4A and later versions
[5:6]	-	-	Reserved, must be 0
4	FUN_PARA_FLAG_DISABLE_SEND_OK	0	Disable Socket SEND_OK interrupt, supported by 0x4A and later versions
3	FUN_PARA_FLAG_SOCKET_CLOSE	0	Socket off mode, supported by 0x46 and later versions.
2	-	-	Reserved, must be 0
1	FUN_PARA_FLAG_TCP_SERVER	0	TCP server multi-connection mode enable bit, supported by 0x44 and later versions.
0	-	-	Reserved, must be 0

① Fun_para_flag_TCP_server: TCP server multi-connection mode enable bit, this bit is 1, TCP server can connect multiple clients, please refer to 9.2.6, Global Configuration for usage, which is valid for all Socket.

② FUN_PARA_FLAG_SOCKET_CLOSE: This parameter is mainly used for TCP, and this bit is 0, which indicates that CH395 will actively close the Socket after the interrupt of SINT_STAT_DISCONNECT or SINT_STAT_TIM_OUT occurs. This bit is 1, which means that the Socket is closed by an external MCU after the above 2 interrupts are generated. Closing the Socket means that CH395 will clear the internal buffer of the Socket and some related variables. In some applications, if CH395 closes the Socket by itself, the residual data may be forcibly cleared. In this case, you can set this bit to 1, read the data by the MCU, and then close the Socket.

③ FUN_PARA_FLAG_DISABLE_SEND_OK: Disable SINT_STAT_SEND_OK interrupt. This bit is 1 and CH395 will no longer generate SINT_STAT_SEND_OK interrupt.

④ FUN_PARA_FLAG_UART_FAST_DEAL: For the serial command with data output, when this bit is 0, CH395 will output the data after receiving the data at an interval of 4 bytes of serial data timeout +TSC time; When this bit is set to 1, CH395 replies the data only at TSC time after receiving the data.

⑤ Fun _ para _ flag _ UART _ flow _ control: serial port flow control enable bit, only applicable to CH395F. When this bit is set to 1, CTS and RTS pins of CH395F will be used as hardware flow control signals to realize the hardware flow control mechanism of serial communication.

⑥ Note that the CMD_SET_FUN_PARA command must be set before initialization, and once initialization is completed, it cannot be set at will. In addition, you need to pay attention to the version number of the parameter.

5.53 CMD_SET_KEEPALIVE_IDLE

This command is used to set the Keepalive idle time, and a 4-byte time value in ms is required. This command is only used for TCP connections. Keepalive idle time refers to the time from the TCP connection without data transmission and reception to the sending of Keepalive packets. The default value is 20000.

5.54 CMD_SET_KEEPALIVE_INTVL

This command is used to set the Keepalive timeout, and a 4-byte time value in ms is required. This command is only used for TCP connections. Keepalive timeout refers to the time waiting for reply after Keepalive packet is sent. The default value is 15000.

5.55 CMD_SET_KEEPALIVE_CNT

This command is used to set the Keepalive timeout, and you need to enter a timeout of 1 byte. This command is only used for TCP connections. Keepalive timeout refers to the maximum number of consecutive unanswered Keepalive packets. The default value is 9.

Assume that the Keepalive IDLE time is idle, the timeout time is INTVL, and the timeout number is CNT.

If the Keepalive function is enabled for the Socket connection, when the TCP connection is idle (no data is sent or received), the Socket will start sending Keepalive packets. If the remote end replies with an ACK in the INTVLms, the connection is considered normal. Otherwise, the Socket thinks that it has timed out after INTVLms, and starts sending Keepalive packets again. If no ACK packets are received within CNT times, it is considered that the current connection is disconnected, and a SINT_STAT_TIM_OUT or SINT_STAT_DISCONNECT interrupt will be generated.

If three variables, IDLE, INTVL and CNT, need to be configured, it should be noted that IDLE must be greater than INTVL and both are multiples of 500.

5.56 CMD_SET_KEEPALIVE_SN

This command is used to turn on or off the Keepalive function of Socket, and it needs to input a 1-byte Socket index and a 1-byte configuration value, where the configuration value is 0 to turn off the Keepalive function of Socket and 1 to turn on it. The default is off.
When the Socket is a TCP client, use this command to open the Keepalive function after creating the Socket.
When the Socket is a TCP server, use this command to open the Keepalive function after generating SINT_STAT_CONNECT.

6. Function Description

6.1 MCU Communication Interface

There are 3 types of communication interfaces between CH395 and the MCU: 8-bit parallel interface, SPI synchronous serial interface, and asynchronous serial interface. When the chip is powered on and reset, CH395 will sample the status of the SEL and TXD pins, and select the communication interface according to the combination of these 2 pin states, refer to the table below (X in the table means not caring about this bit, 0 means low level, and 1 means high level or floating). SEL and TXD pins have built-in pull-up resistors, and the asynchronous serial port is selected by default. The desired communication interface can be selected by directly grounding the SEL or TXD pin to set to low.

Table 6-1 Interface selection

SEL pin	TXD pin	Select the communication interface
1	1	UART
1	0	SPI interface
0	1	8-bit parallel
0	0	Error interface

The interrupt request output of the INT# pin of the CH395 chip is valid for low by default. It can be connected to the interrupt input pin or ordinary input pin of the MCU. The MCU can use the interrupt method or query method to obtain the interrupt request of the CH395.

6.2 SPI Serial Interface

SPI synchronous serial interface signal line includes SPI chip selection input pin SCS, serial clock input pin SCK, serial data input pin SDI and serial data output pin SDO. Through SPI interface, CH395 can be connected to SPI serial bus of various MCU, DSP and MCU with less wires.

The SPI interface of CH395 supports SPI mode 0 and SPI mode 3. CH395 always inputs data from the rising edge of SPI clock SCK, and outputs data from the falling edge of SCK when output is allowed. The order of data bits is MSB first, and 8 bits are counted as a word.

6.2.1 SPI Operation Steps

The operation steps of SPI are as follows:

① MCU generates SPI chip selection of CH395 chip, active low;

② MCU sends out a byte of data according to SPI output mode. CH395 always takes the first byte received after SPI chip selects SCS as the command code and the subsequent bytes as the data.

③ MCU delays TSC time (about 0.6us) to wait for SPI interface of CH395 to be idle;

④ If it is a write operation, the MCU sends a byte of data to be written to CH395. After the SPI interface is idle, the MCU continues to send several bytes of data to be written, and CH395 receives them in turn until the MCU prohibits SPI chip selection.

⑤ If it is a read operation, the MCU receives one byte of data from CH395, and after the SPI interface is idle, the MCU continues to receive several bytes of data from CH395 until the MCU prohibits SPI chip selection;

⑥ MCU prohibits SPI chip selection of CH395 chip to end the current SPI operation.

The following figure is the logic timing diagram of SPI interface. Figure 6-1 shows issuing a command for 12H and writing data for 34H, and Figure 6-2 shows issuing a command for 28H and reading data for 78H.

Figure 6-1 Example 1

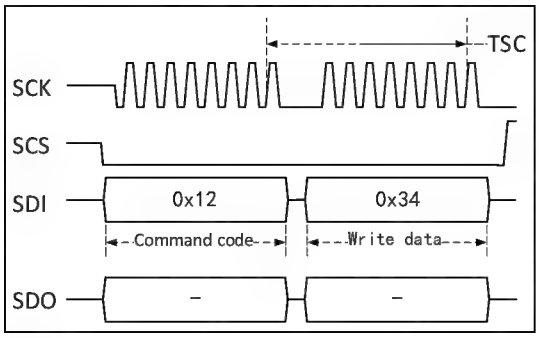
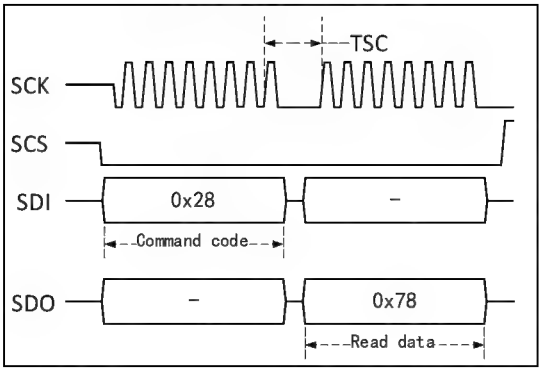


Figure 6-2 Example 2



6.3 UART

UART signal line includes a serial data input pin RXD and a serial data output pin TXD. Through the serial interface, CH395 can connect with MCU, DSP and MCU point to point with the least number of wires.

CH395F supports serial port hardware flow control (clearing the sending input pin CTS and sending request output pin RTS) and RS485 transceiver automatic switching (sending status output pin TNOW). By default, the flow control function is disabled, but it can be enabled by the CMD_SET_FUN_PARA command to realize the hardware flow control mechanism.

The serial data format of CH395 is a standard byte transmission mode, including 1 start bit, 8 data bits and 1 stop bit.

CH395 not only supports the hardware to set the default serial communication baud rate, but also supports the MCU to select the appropriate communication baud rate at any time through the cmd_set_baud command. After each power-on reset, the default baud rate of CH395 serial communication is set by the level combination of SDO, SDI and SCK, refer to the following table (0 in the table stands for low level, 1 stands for high level or floating). SDO, SDI and SCK pins are suspended, and 9600bps is selected by default.

The baud rate error of CH395 serial port transmitting signal is less than 0.3%, and the allowable baud rate error of serial port receiving signal is less than 2%.

Table 6-2 Baud rate selection

SDO pin	SDI pin	SCK pin	Default baud rate of serial communication after power-on reset.
1	1	1	9600bps
1	1	0	57600bps
1	0	1	115200bps
1	0	0	460800bps
0	1	1	250000bps
0	1	0	1Mbps
0	0	1	2Mbps
0	0	0	921600bps

In order to distinguish the command code from the data, CH395 requires the MCU to send 2 synchronization code bytes (57H and ABH) through the serial port first, then send the command code, and then send or receive the data. CH395 will check the interval time between the above 2 synchronization code bytes and between the synchronization code and the command code. If the interval time is longer than the serial input timeout time SER_CMD_TIMEOUT (about 40ms), CH395 will discard the synchronization code and the command packet.
For the serial command with returned data, CH395 will return the data after about TE7 time (see Table 7-4) after receiving the command.

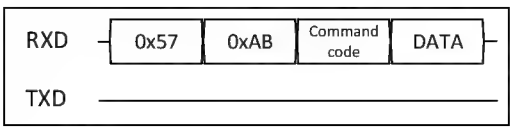
6.3.1 Serial Port Operation Steps

The serial port command operation steps are as follows:

① The MCU sends the first synchronization code 57H to CH395 through the serial port;
② The MCU issues the second synchronization code ABH to CH395;
③ The MCU issues command code to CH395;
④ If the command has input data, input data is sent to CH395 in turn, one byte at a time;
⑤ If the command has output data, the output data is received from CH395 in turn, one byte at a time.

The following figure is the logical timing diagram of the serial port.

Figure 6-3 Example 3



6.4 8-bit Parallel Interface

The parallel signal lines include: 8-bit bidirectional data bus D7 to D0, read gate input pin RD#, write gate input pin WR#, chip select input pin PCS#, and address input pin A0. The PCS# of the CH395 chip is used to select devices when the MCU has multiple peripheral devices. Through the passive parallel interface, the CH395 chip can be easily attached to the system bus of various 8-bit MCUs, DSPs, and MCUs, and can coexist with multiple peripheral devices.
For MCUs similar to Intel parallel timing, the RD# pins and WR# pins of the CH395 chip can be connected to the read gate output pin and the write gate output pin of the MCU respectively. For MCUs similar to Motorola parallel timing, the RD# pin of the CH395 chip should be connected to the low level, and the WR# pin is connected to the MCU's read and write direction output pin R/-W.

The following table is the truth table for parallel I/O operations (X in the table means that this bit is not caring about, Z means that CH395 three-state prohibited).

Table 6-3 Parallel operation truth table

PCS#	WR#	RD#	A0	D7-D0	Practical operation of CH395 chip
1	X	X	X	X/Z	CH395 is not selected and no operation is performed.
0	1	1	X	X/Z	Although selected, no operation is performed and no operation is performed.
0	0	1/X	1	Input	Write command code to the command port of CH395.
0	0	1/X	0	Input	Write data to the data port of CH395.
0	1	0	0	Output	Read data from the data port of CH395.
0	1	0	1	Output	Read interface status from the command port of CH395: Bit 7 is the interrupt flag, low valid, equivalent to the INT# pin.

6.4.1 Parallel Operation Steps

When the CH395A0 pin is high, select the command port, and can write a new command, or read the interface status; when the A0 pin is low, select the data port, and can read and write data.

The MCU reads and writes the CH395 chip through an 8-bit parallel port. All operations are composed of a command code, several input data and several output data. Some commands do not require input data, and some commands do not output data. The command operation steps are as follows:

① When A0=1, the MCU writes command code to the command port;

② The MCU delays TSC time (about 0.6us) to wait for the parallel port of CH395 to be free;

③ If the command has input data, write the input data one byte at a time when A0=0;

④ If the command has output data, the output data is read one byte at a time when A0=0.

6.5 Other Hardware

CH395 chip integrates 10M/100M Ethernet PHY, MAC, serial port, SPI-Slave controller, passive parallel interface, SRAM, high-speed MCU and PLL frequency multiplier, power-on reset circuit, etc.

CH395 chip has built-in power-on reset circuit, which can also be controlled by pulling RSTI pin low. RSTI pin is used to input asynchronous reset signal from outside; When the RSTI pin is low, the CH395 chip is reset; When the RSTI pin returns to the high level, CH395 will enter the initialization phase for about 15ms, during which the host prohibits the operation of CH395.

7. Parameters

7.1 Absolute Maximum Ratings

(Critical or exceeding the absolute maximum value will probably cause the chip to work improperly or even be damaged)

Table 7-1 Absolute maximum value

Name	Parameter description	Min.	Max.	Unit
T_A	Ambient temperature during operation	-40	85	℃
T_J	Junction temperature range	-40	100	℃
T_S	Stored ambient temperature	-55	150	℃
V_{CC33}	Operating power supply voltage	-0.4	4.0	V
V_{CCIO}	I/O supply voltage	-0.4	4.0	V
V_{DDK}	Power decoupling end of core analog circuit	-0.4	1.5	V
V_{ETH}	Voltage on ETH physical signal pin	-0.4	V_{CC33}+0.4	V
V_{IN}	Input voltage on other I/O pins	-0.4	V_{CCIO}+0.4	V
$V_{ESD(HBM)}$	ESD electrostatic discharge voltage of common I/O pin (HBM)	6K		V
I_{IO}	Sinking current on I/O pin		20	mA
	Output current on I/O pin		20	

7.2 Electrical Parameters

Table 7-2 Electrical Parameter (V_{CC33} = 3.3V， V_{CCIO} = 3.3V， T_A = 25℃)

Name	Parameter description		Min.	Typ.	Max.	Unit
V_{CC33}	Operating power supply voltage		3.2	3.3	3.4	V
V_{CCIO}	I/O supply voltage		1.7	3.3	3.6	V
V_{IL}	I/O pin, input low voltage.	V_{DDIO} = 3.3V	0		0.8	V
		V_{DDIO} = 1.8V	0		0.6	
V_{IH}	I/O pin, input high voltage.	V_{DDIO} = 3.3V	2.0		V_{CCIO}	V
		V_{DDIO} = 1.8V	1.2		V_{CCIO}	
V_{OL}	Low level output voltage, single pin sinks 5mA current			0.4	0.6	V
V_{OH}	High level output voltage, single pin output 5mA current		V_{CCIO}-0.6	V_{CCIO}-0.4		V
V_{hys}	Voltage hysteresis of I/O Schmitt trigger			150		mV
C_{IO}	I/O pin capacitance			5		pF
R_{PU}	Pull up equivalent resistance		30	40	55	kΩ
R_{PD}	Pull down equivalent resistance		30	40	55	kΩ

7.3 Power Consumption

Table 7-3 Power consumption (V_{CC33} = 3.3V， V_{CCIO} = 3.3V， T_A = 25℃)

Symbol	Parameter	Condition	Typical	Unit

		(All current, with network regulator)		
I_{DD0}	Supply current in transfer status	100BASE-TX channel link is successful and there is a packet on the transceiver channel	74.9	mA
		10BASE-TX channel link is successful and there is a packet on the transceiver channel	46.8	
I_{DD1}	Supply current in transfer status	100BASE-TX channel link is successful and there is no packet on the transceiver channel	74.9	mA
		10BASE-TX channel link is successful and there is no packet on the v channel	47.5	
I_{DD2}	Supply current in disconnect status	Both the 100BASE-TX and 10BASE-TX paths are not successfully linked and the PHY is in auto-negotiation.	54.4	mA
I_{DD3}	Sleep power consumption		380	uA

7.4 Control Timing Parameters

Table 7-4 Control timing parameters

Name	Parameter description	Condition	Typ.	Unit
TWAK	Wake-up time exiting from low power state		100	us
TE0	Execution time of CMD_RESET_ALL command		15	ms
TE1	Execution time of CMD_INIT_CH395 command		5	ms
TE2	Execution time of CMD00_EEPROM_ERASE command		21.1	us
TE3	Execution time of CMD30_EEPROM_WRITE command.		6.5	us
TE4	Execution time of CMD30_EEPROM_READ command.		5.2	us
TE5	Execution time of CMD_SET_MAC_ADDR command.		30	us
TE6	Execution time of the remaining commands		1.5	us
TE7	Return time of command data in serial mode		See note 1	-
TE8	Execution time of cmd_set_baud command.		See note 2	
TUC	Rrequired interval between transmission command code and data in serial mode		0	us
TUD	Required interval between data bytes in serial mode		0	us
TSC	Required interval between transmission command code and data in SPI or parallel port mode	Read-write transceiver buffer	See table 7-5	us
		Other orders	0.6	
TSD	Required interval between data bytes in SPI or parallel port mode	Read-write transceiver buffer	See table 7-5	us
		Other orders	0.3	

Note: 1. For serial commands with data output, CH395 will interval TE7 time after receiving data before outputting data.TE7 time is determined by bit7 of CMD_SET_FUN_PARA command. When the default is 0, the TE7 time is about the timeout time of 4 bytes of serial data plus the TSC time, which is calculated as 4(10/BaudRate) + TSC (unit: s). For example, when the baud rate is 921600, the timeout time is about 4*(10/921600)+0.0000006≈0.000044 seconds (44us); when bit7 is 1, the return time is only TSC time (0.6us).*

2. TE8 time is TE7+1ms.

In SPI or parallel port mode, CH395 only needs to ensure the interval of TSL time between the length and data, and all other bytes need no interval.

Table 7-5 Read and write transceiver buffer timing parameters

Name	Parameter description	Condition	Typ.	Unit
TSL	The required interval between transmission length and data in SPI or parallel port mode	Read-receive buffer	0.4	us
		Write-send buffer	0.2	

Figure 7-1 SPI mode write-send buffer

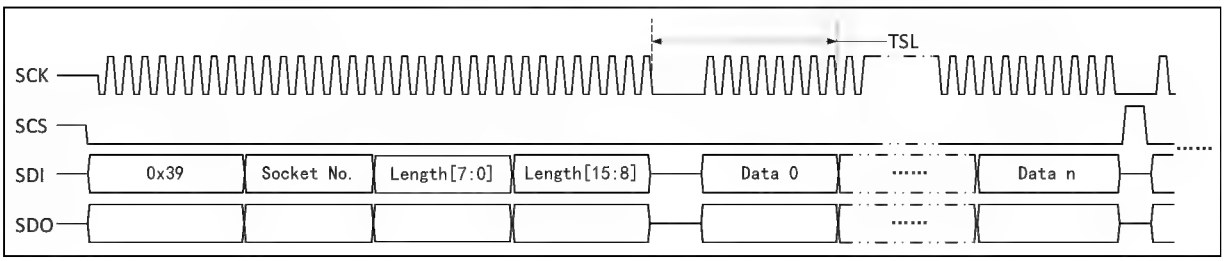


Figure 7-2 SPI mode read-receive buffer

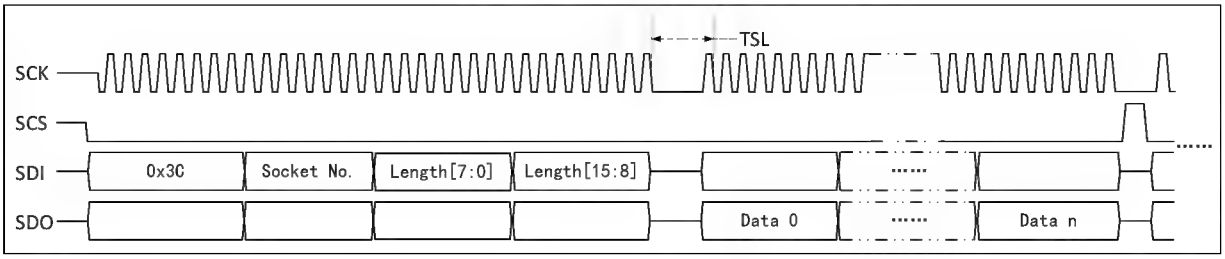


Figure 7-3 Parallel port mode write-send buffer

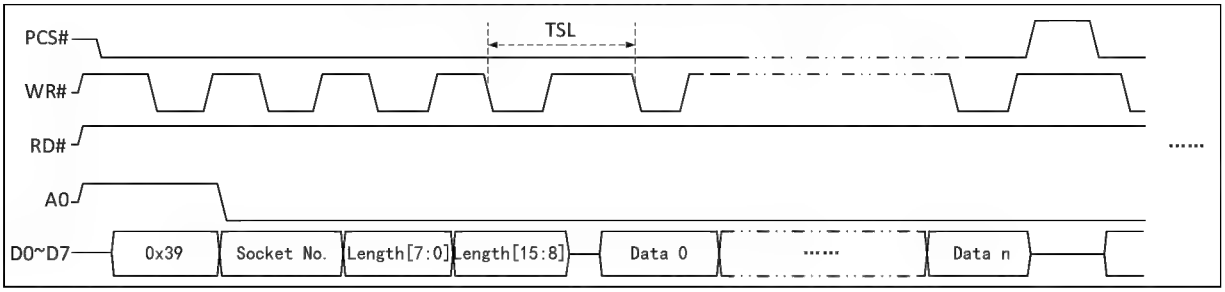
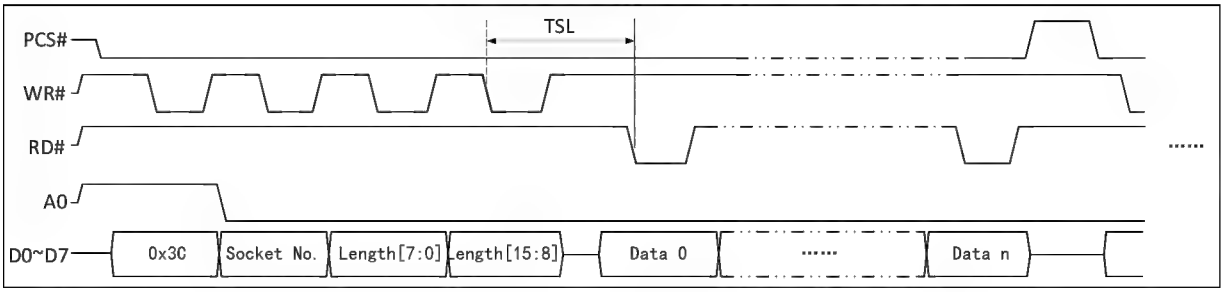


Figure 7-4 Parallel port mode write-send buffer



7.5 AC Electrical Characteristic and Timing

7.5.1 SPI Timing

Figure 7-5 SPI mode 0 timing figure

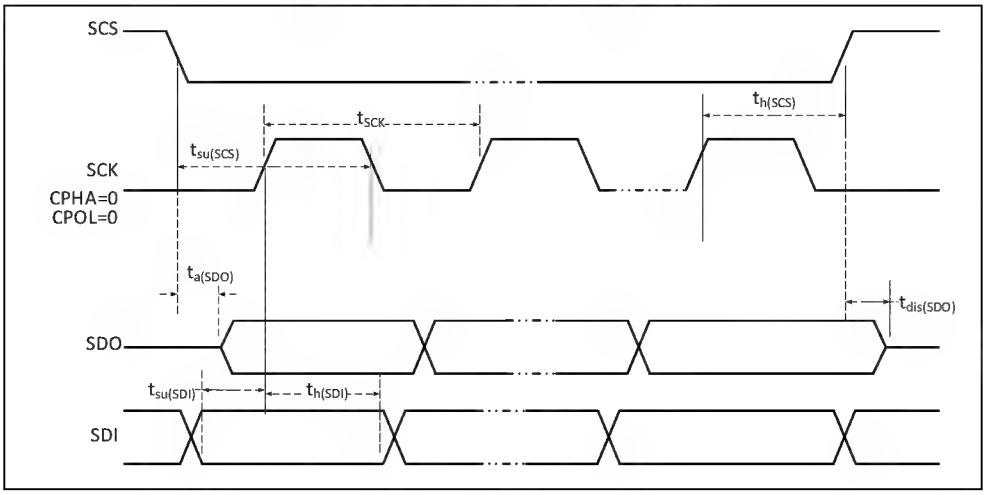


Figure 7-6 SPI mode 3 timing figure

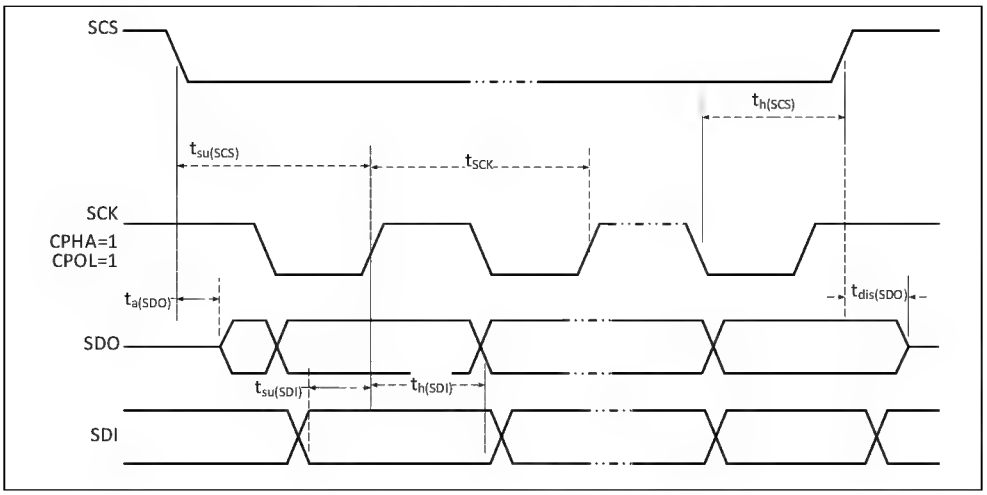


Table 7-6 SPI parameter table (V_{CC33} = 3.3V， V_{CCIO} = 3.3V， T_A = 25℃)

Symbol	Parameter	Condition	Min.	Max.	Unit
f_{SCK}	SCK clock frequency of SPI			40	MHz
$t_{SU(SCS)}$	SCS setup time		17		ns
$t_{h(SCS)}$	SCS holding time		17		ns
$t_{SU(SDI)}$	Data input setup time		4		ns
$t_{h(SDI)}$	Data input holding time		2		ns
$t_{a(SDO)}$	Data output access time		0	8	ns
$t_{dis(SDO)}$	Data output disable time		0	10	ns

7.5.2 Parallel Port Timing

Figure 7-7 Parallel port timing figure

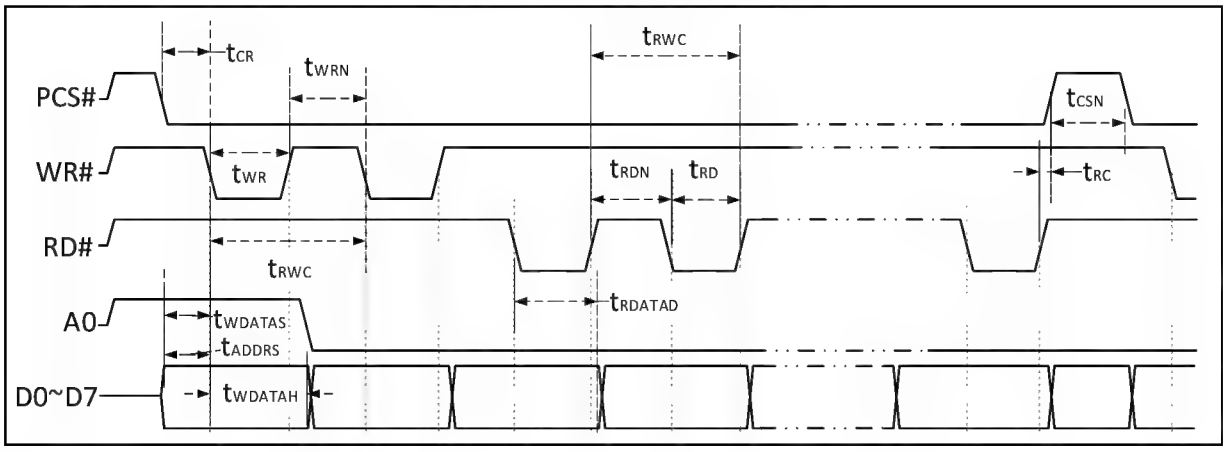


Table 7-7 Parallel port parameter table (V_{CC33} = 3.3V， V_{CCIO} = 3.3V， T_A = 25℃)

Symbol	Parameter	Condition	Min.	Max.	Unit
t_{CR}	PCS# setup time		1		ns
t_{RC}	PCS# holding time		0		ns
t_{ADDRS}	Address setup time		1		ns
t_{WDATAS}	Data input setup time		1		ns
t_{WDATAH}	Data input holding time		10		ns
T_{RDATAD}	Data output delay time		15		ns
t_{RDN}	RD# high level holding time		25		ns
t_{RD}	RD# low level holding time		30		ns
t_{WRN}	WR# high level holding time		25		ns
t_{WR}	WR# low level holding time		30		ns
t_{RWC}	Read-write cycle time		70		ns
t_{CSN}	CS high level holding time		35		ns

7.5.3 Oscillator and Crystal Oscillator Timing

Table 7-8 Parameters table of oscillator and crystal oscillator timing

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
TCKF	Crystal frequency	Recommended within 30ppm	24.999	25	25.001	MHz
TPWH	High clock pulse width		15	20	25	ns
TPWL	Low clock pulse width		15	20	25	ns

Note: The XI and XO pins have built-in two oscillation capacitors required by an external crystal with a load capacitor of 12pF, and only the crystal is needed externally; If an external crystal with a load capacitance of 20pF is selected, XI and XO need an additional oscillation capacitance of 15pF to the ground respectively.

7.5.4 Reset Timing

Table 7-9 Reset timing parameter table

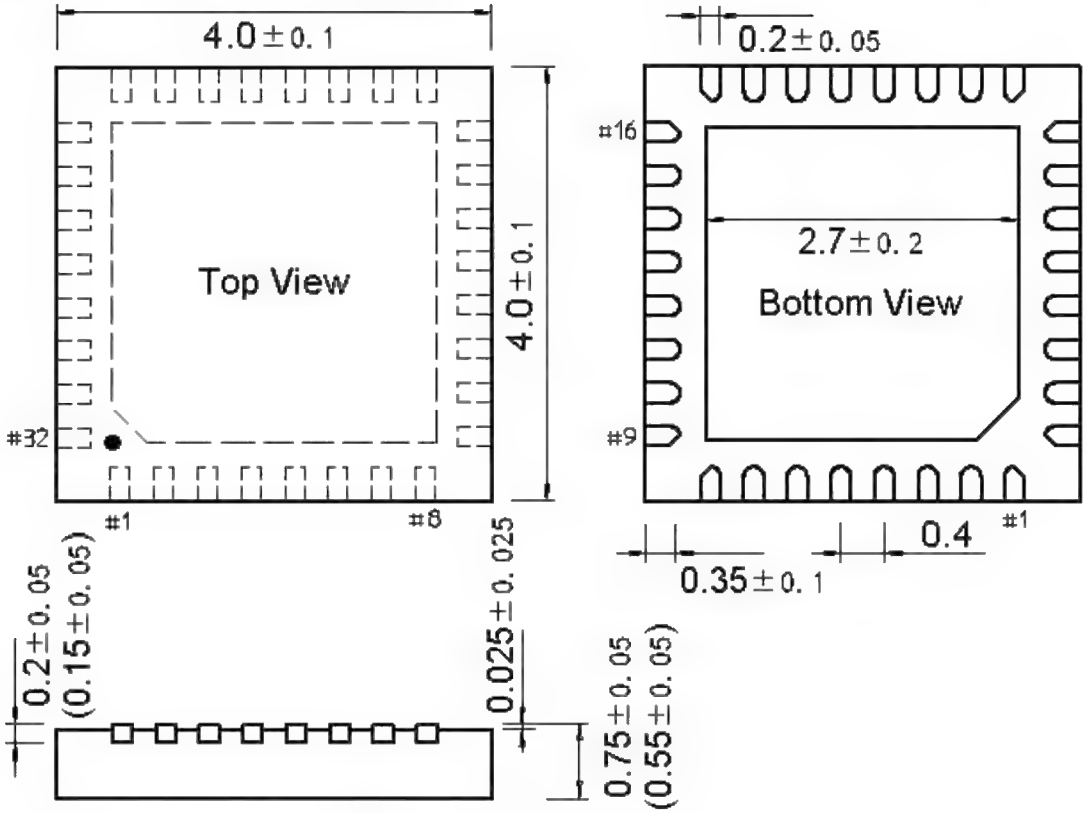
Symbol	Parameter	Min.	Typ.	Max.	Unit
$t_{RSTTEMPO}$	RSTI low level width	1			us
$t_{RSTTEMP1}$	RSTI high to host operable	12	15	19	ms
$t_{RSTTEMP2}$	Reset command valid to the host for operation	12	15	19	ms
$t_{RSTTEMP3}$	Power on and reset to the host for operation	27	30	35	ms

8. Package Information

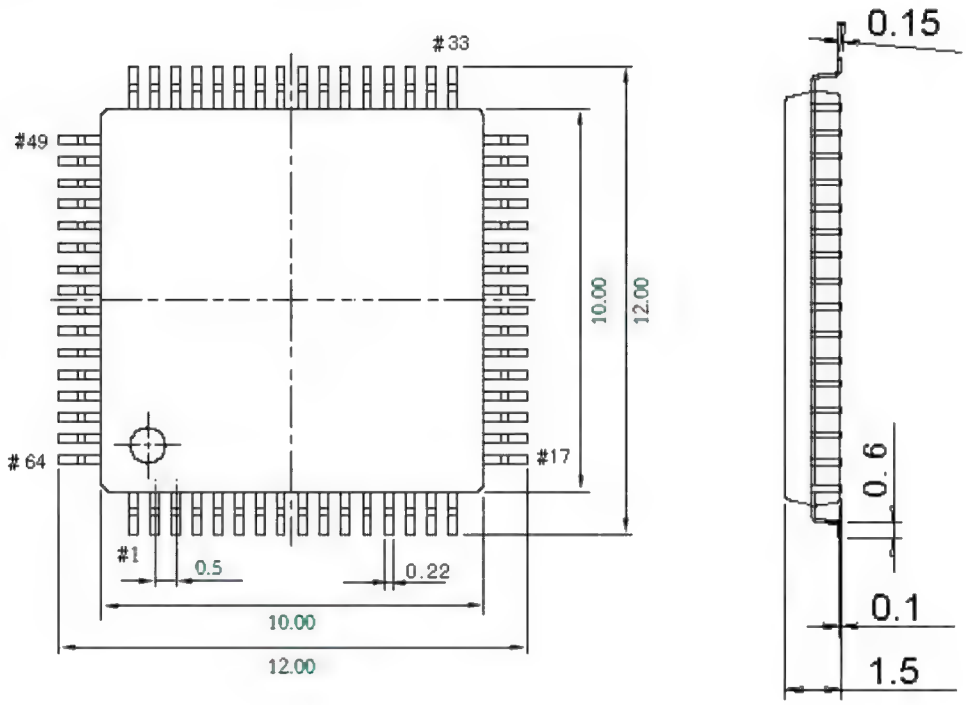
Note: All dimensions are in millimeters.

Pin center spacing is nominal with no error, except for dimensional error of no more than ±0.2mm.

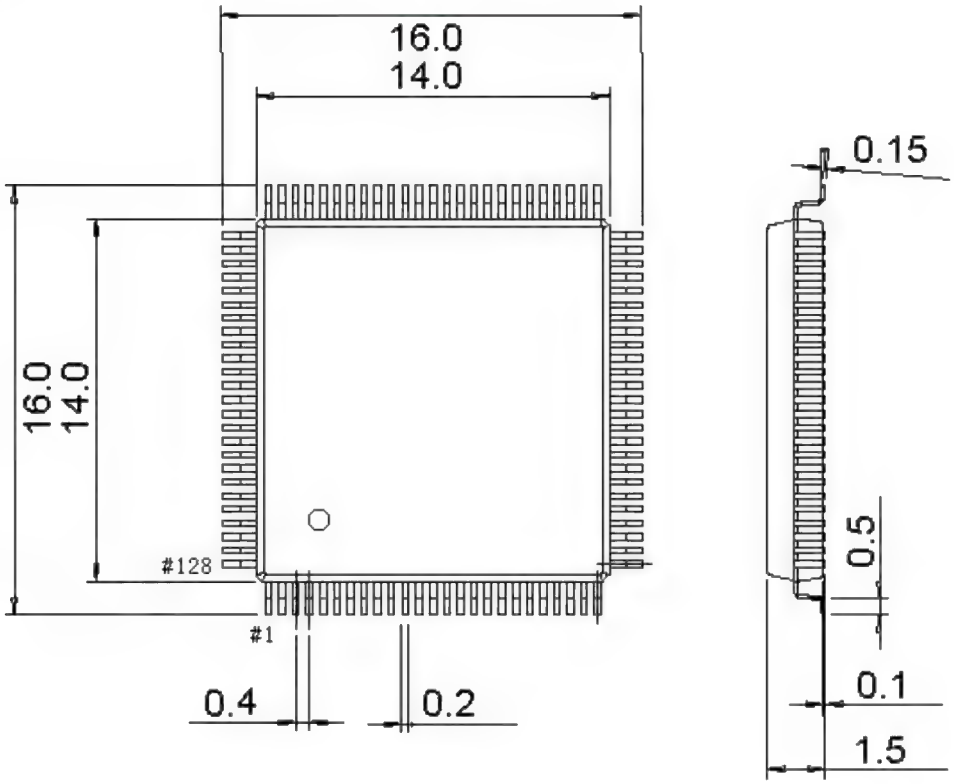
8.1 QFN32



8.2 LQFP64M



8.3 LQFP128



9. Application

9.1 Application Base

CH395 integrates IPv4, ARP, ICMP, UDP, TCP and other protocols.

TCP and UDP are two important transport layer protocols, both of which use IP as the network layer protocol.

TCP is a connection-oriented transmission, which can provide reliable byte stream transmission service.

UDP is a simple datagram-oriented transport layer protocol. Unlike TCP, UDP can't guarantee that the datagram can reach its destination accurately.

TCP provides high-reliability communication for network devices, and is responsible for transmitting application-layer data to network layer in segments, acknowledging receiving packets, setting timeout and other functions, thus realizing end-to-end reliable transmission, so that the application layer does not need to pay attention to the underlying details. In contrast, UDP provides a simple and fast service, which is only responsible for transmitting datagrams from one network terminal to another, and does not guarantee the delivery of datagrams. The reliability needs to be realized by the application layer itself. Both TCP and UDP data are transmitted through the IP protocol of the network layer.

ICMP is an accessory protocol of IP protocol, which is used by IP layer to exchange error messages or other important information with other hosts or routers. For example, when CH395 generates unreachable interrupt, it is through ICMP that error message switching is performed. PING also uses the ICMP protocol.

ARP is an address resolution protocol, which is used to convert the addresses used by IP layer and network interface layer.

9.2 Application Reference Steps

This chapter introduces the operation process of transmitting and receiving data, and can refer to the routine for details.

9.2.1 Initialize CH395, Necessary Operations

① Send the command CMD_SET_MAC_ADDR to set the MAC address of CH395;
② Send the command CMD_SET_IP_ADDR to set the IP address of CH395;
③ Send the command CMD_SET_GWIP_ADDR to set the gateway IP address of CH395;
④ Send the command CMD_SET_MASK_ADDR to set the subnet mask of CH395;
⑤ Send the command CMD_INIT_CH395 to initialize CH395;
⑥ Delay 2ms or more to send the command CMD_GET_CMD_STATUS to get the execution status of CMD_INIT_CH395, if it returns CH395_ERR_BUSY it means that CH395 is executing the command internally and needs to be executed again ⑥; if it returns CH395_ERR_SUCCESS, it means that the command has been executed successfully.CMD_INIT_CH395 normally takes 5ms to be executed. INIT_CH395 normally takes 5ms to execute.

Step ① is generally not needed, the CH395 has already burned the MAC address assigned by IEEE when it is shipped from the factory.
If you need to start DHCP, steps ② - ④ are not required.
Step ④ is optional, and the default subnet mask of 255.255.255.0 is generally not required.

After receiving the cmd_init_CH395 command, Ch395 initializes the internal TCP/IP protocol stack, and initializes the MAC and PHY. At this time, the MAC and PHY will be initialized to the auto-negotiation mode. If PHY needs to work in other modes, such as 10M full-duplex mode, it needs to send the CMD_SET_PHY command for setting after the successful execution of CMD_INIT_CH395.

9.2.2 Initialize Socket to MACRAW Mode

The initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set the Socket to work in MACRAW mode;
② Send the command CMD_OPEN_SOCKET_SN to open Socket;
③ Send the command CMD_GET_CMD_STATUS with a delay of more than 2ms to get the execution status of CMD_OPEN_SOCKET_SN. If the command is being executed in CH395_ERR_BUSY, it needs to be executed again. If CH395_ERR_SUCCESS is returned, the command is executed successfully. Other values indicate that opening the Socket failed.

IEEE802.3 Ethernet frame format:

Destination MAC	Source MAC	Type	Data	CRC32
6Byte	6Byte	2Byte	46-1500Byte	4Byte

In MACRAW mode, CH395 will transparently transmit the data between Ethernet and MCU, and will not encapsulate the data with TCP/IP. when CH395 receives the data, it will check the Ethernet redundancy check CRC32, and if the check is wrong, the packet will not be forwarded to the MCU. when CH395 sends the data, it will add the Ethernet redundancy check CRC32 at the end of the packet. The length of the data written by the MCU to the CH395 each time shall not be greater than 1514, and the CH395 will encapsulate the data written by the MCU each time into a frame of data to be sent. When CH395 receives data from the Ethernet, it will notify the MCU, and then the MCU should immediately read all the data away from the internal receive buffer of CH395.

Only Socket0 can set this mode, and other Sockets will not be available.

9.2.3 Initialize Socket to IPRAW Mode

The initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set the Socket to work in IPRAW mode;
② Send the command CMD_SET_IP_ADDR_SN to set the destination IP address;
③ Send the command CMD_SET_IPRAW_PRO_SN to set the protocol field;
④ Send the command CMD_OPEN_SOCKET_SN to open Socket;
⑤ Send the command CMD_GET_CMD_STATUS with a delay of more than 2ms to get the execution status of CMD_OPEN_SOCKET_SN. If the command is being executed in CH395_ERR_BUSY, it needs to be executed again. If the return of CH395_ERR_SUCCESS indicates that the command was executed successfully, other values indicate that opening the Socket failed.

IP message structure:

Destination MAC	Source MAC	Type	IP header	IPRAW data	CRC32
6Byte	6Byte	2Byte	20Byte	Max.1480Byte	4Byte

After the MCU writes several bytes of data streams to CH395, CH395 encapsulates the protocol field of this Socket in the IP header and encapsulates the data stream in the IPRAW data part for sending. The maximum

length allowed for a IPRAW packet to send is 1480 bytes. If the length of the data stream written by the MCU is greater than 1480 bytes, CH395 will encapsulate the data stream into several IP packets for transmission, and after each packet is successfully sent, a SINT_STAT_SEND_OK interrupt will occur. The byte length written by the MCU each time must not be greater than the length of the sending buffer, and the next data writing can be performed only after receiving the SINT_STAT_SENBUF_FREE interrupt. If a SINT_STAT_TIM_OUT interrupt occurs, it means that the data transmission fails, and there are generally two reasons for the failure to send data:

① If the destination IP address and CH395 are on the same subnet, the network device with the destination IP address may not be online.

② If the destination IP address and CH395 are not on the same subnet, the gateway of CH395 may not be online.

When CH395 receives the IP packet, it first checks whether the protocol field and the protocol field set by Socket are the same. If the same, copy the IPRAW packet to the receiving buffer and generates a SINT_STAT_RECV interrupt. After the MCU receives this interrupt, it can send the command CMD_GET_RECV_LEN_SN to obtain the length of the received buffer data, and then send the command CMD_READ_RECV_SN to read the buffer data. The MCU can read all data at once or in multiple times. Since CH395 cannot be flow-controlled in IPRAW mode, it is recommended that the MCU read all data immediately after querying the received data interrupt port to avoid being overwritten by subsequent data.

Notes on the Protocol Field Setting

CH395 has higher priority in handling IPRAW than UDP and TCP. If the IP protocol field is set to 17 (UDP) or 6 (TCP), there may be a possibility of conflict with other Sockets. Pay attention to avoid it when using it. The following two situations are listed for explanation:

① Socket0 is set to IPRAW mode, IP protocol field is 17, and Socket1 is UDP mode. In UDP mode, the protocol field of IP packet is also 17, which will cause the data communicated by Socket1 to be intercepted by Socket0 and unable to receive the data.

② Socket0 is set to IPRAW mode, IP protocol field is 6, and Socket1 is set to TCP mode. In TCP mode, the protocol field of IP packet is also 6, which will cause the data communicated by Socket1 to be intercepted by Socket0 and unable to receive the data.

9.2.4 Initialize Socket to UDP Mode

The initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set the Socket to work in UDP mode;

② Send the command CMD_SET_IP_ADDR_SN to set the destination IP address;

③ Send the command CMD_SET_DES_PORT_SN to set the destination port;

④ Send the command CMD_SET_SOUR_PORT_SN to set the source port;

⑤ Send the command CMD_OPEN_SOCKET_SN to open Socket;

⑥ Send the command CMD_GET_CMD_STATUS with a delay of more than 2ms to get the execution status of CMD_OPEN_SOCKET_SN. If the command is being executed in CH395_ERR_BUSY, it needs to be executed again. If the return of CH395_ERR_SUCCESS indicates that the command was executed successfully, other values indicate that opening the Socket failed.

UDP message structure:

Destination MAC	Source MAC	Type	IP header	UDP header	UDP data	CRC32
6Byte	6Byte	2Byte	20Byte	8Byte	Max.1472Byte	4Byte

UDP is a simple and unreliable transport layer protocol for data messages, which has a high transmission speed and cannot guarantee that the data can reach its destination. The application layer must ensure the reliability and stability of transmission.

After the MCU writes several bytes of data streams to CH395, the CH395 data stream is encapsulated in the UDP data part for transmission. The maximum length that a UDP packet can send is 1472 bytes. If the length of the data stream written by the MCU is greater than 1472 bytes, CH395 will encapsulate the data stream into several UDP packets for transmission, and after each packet is successfully sent, a SINT_STAT_SEND_OK interrupt will occur. The byte length written by the MCU each time must not be greater than the length of the sending buffer, and the next data writing can be performed only after receiving the SINT_STAT_SENBUF_FREE interrupt. If data transmission fails, SINT_STAT_TIM_OUT interrupt will occur, resulting in failure to send data. The following two reasons are generally:

① If the destination IP address and CH395 are on the same subnet, the network device with the destination IP address may not be online.

② If the destination IP address and CH395 are not on the same subnet, the gateway of CH395 may not be online.

When CH395 receives the UDP message, it copies the UDP data into the Socket receiving buffer and generates a SINT_STAT_RECV interrupt. After the MCU receives this interrupt, it can send the command CMD_GET_RECV_LEN_SN to obtain the length of the received buffer data, and then send the command CMD_READ_RECV_SN to read the buffer data. Since CH395 cannot provide flow control in UDP mode, it is recommended that the received data be read quickly and quickly to avoid being overwritten by subsequent data.

CH395 supports 2 UDP modes: UDP client and UDP server. The UDP client can only communicate with the specified IP and port, and the UDP server can communicate with any remote IP and port.

There are some differences in the use of the two:

① Initialization step ②, if the destination IP address is 0XFFFFFFFF, this socket will enter UDP server mode, otherwise UDP client mode.

② The MCU reads data from CH395. In client mode, CH395 directly sends the received data stream to the MCU. The MCU can read all the data at once or only read out part of the data. In server mode, CH395 will add an 8-byte information table to the head of the data. The MCU can obtain the source information of the data packet based on the information table. The MCU must read out all the data at one time.

Packet length	Port	IP address	Data
2Byte	2Byte	4Byte	NByte

③ The MCU sends data, and CH395 directly sends the data to the destination IP and port specified during initialization in the client mode. In server mode, CH395 can send data to any IP and port, and the MCU can set the destination IP and destination port before sending.

9.2.5 Initialize Socket to TCP Client Mode

The initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set the Socket to work in TCP mode;

② Send the command CMD_SET_IP_ADDR_SN to set the destination IP address;

③ Send the command CMD_SET_DES_PORT_SN to set the destination port;

④ Send the command CMD_SET_SOUR_PORT_SN to set the source port;

⑤ Send the command CMD_OPEN_SOCKET_SN to open Socket;

⑥ Send the command CMD_GET_CMD_STATUS with a delay of more than 2ms to get the execution status of

CMD_OPEN_SOCKET_SN. If the command is being executed in CH395_ERR_BUSY, it needs to be executed again. If CH395_ERR_SUCCESS is returned, it indicates that the command was executed successfully, and other values indicate that opening the Socket failed;

⑦ Send the command CMD_TCP_CONNECT_SN for TCP connection;

⑧ Delay 2ms or more to send the command CMD_GET_CMD_STATUS to get the execution status of CMD_TCP_CONNECT_SN, if it returns CH395_ERR_BUSY, it means that the command is being executed inside CH395, and it needs to be executed again ⑧; if it returns CH395_ERR_SUCCESS, it means that the command is executed successfully. Other values indicate that the command execution fails. Returning CH395_ERR_SUCCESS only means that the command is executed successfully, it does not mean that the TCP connection is successful. If the TCP connection is successful, CH395 will generate SINT_STAT_CONNECT interrupt. If the connection fails CH395 will generate a SINT_STAT_TIM_OUT interrupt, and if it is necessary to connect again, start execution from ⑤ again.

TCP message structure:

Destination MAC	Source MAC	Type	IP header	TCP header	TCP data	CRC32
6Byte	6Byte	2Byte	20Byte	20Byte	Max. 1460Byte	4Byte

TCP provides a connection-oriented and reliable byte stream service.

CH395 generates SINT_STAT_CONNECT, which indicates that TCP connection is established and data can be sent and received. Data transmission operation is not allowed before the connection is established.

After the MCU writes several bytes of data stream to CH395, the CH395 data stream is encapsulated in the TCP data part and sent. The maximum length that a TCP packet can send is TCP MSS bytes. If the length of the data stream written by the MCU is longer than TCP MSS bytes, CH395 will package the data stream into several TCP packets for sending, and each packet will generate a SINT_STAT_SEND_OK interrupt after it is successfully sent. The length of bytes written by the MCU at a time shall not be greater than the length of the sending buffer, and the next data writing operation can only be carried out after receiving the interrupt of SINT_STAT_SENBUF_FREE. In TCP mode, if the data transmission fails, the SINT_STAT_TIM_OUT interrupt will be generated, and CH395 will automatically close this Socket (refer to 5.51 when FUN_PARA_FLAG_SOCKET_CLOSE is 0).

When CH395 receives the TCP message, it copies the TCP data into the Socket receiving buffer and generates a SINT_STAT_RECV interrupt. After the MCU receives this interrupt, it can send the command CMD_GET_RECV_LEN_SN to obtain the length of the received buffer data, and then sends the command CMD_READ_RECV_SN to read the buffer data. The MCU can read all the data at once or only part of the data. The remaining space in the receiving buffer is the TCP window. After each time the MCU reads the data, CH395 will check the remaining space in the receiving buffer and inform the TCP server of the current window size.

9.2.6 Initialize Socket to TCP Server Mode

The initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set the Socket to work in TCP mode;

② Send the command CMD_SET_SOUR_PORT_SN to set the source port Sport;

③ Send the command CMD_OPEN_SOCKET_SN to open Socket;

④ Send the command CMD_GET_CMD_STATUS with a delay of more than 2ms to get the execution status of CMD_OPEN_SOCKET_SN. If the command is being executed in CH395_ERR_BUSY, it needs to be executed again. If the return of CH395_ERR_SUCCESS indicates that the command was executed successfully, other values indicate that opening the Socket failed.

In the TCP server mode, if the client connects, the Socket will always be in listening state, and there will be no timeout interruption. If the TCP connection is successful, CH395 will interrupt SINT_STAT_CONNECT. At this time, the MCU can obtain the IP address and port number of the client by sending the command CMD_GET_REMOT_IPP_SN.

By default, the server multi-connection function is turned off. In TCP server mode, only one TCP connection can be established for each Socket.

When the multi-connection mode is enabled, the TCP server can connect multiple TCP connections, and the MCU needs to set the source port of the Socket to be consistent with the source port of the server. If the TCP server listens to the connection, CH395 will find out whether the source ports of all the Sockets are consistent with the current server, and the protocol type is TCP, and it is closed. If it is found, it will immediately open the Socket, assign the connection to the socket, and notify the MCU that there is a connection event. If it is not found, it will reset the connection. In this mode, the Socket of the server is only used for monitoring, and the MCU needs to allocate other sockets for the connection of the server. For example, Socket0 is set to server mode, and Socket1 and Socket2 are used to connect this server. The steps are as follows:
Socket0 executes ①-④;

⑤ Send command CMD_SET_SOUR_PORT_SN to Socket1 to set source port Sport;
⑥ Send command CMD_SET_PROTO_TYPE_SN to Socket1 to set Socket working in TCP mode;
⑦ Send command CMD_SET_SOUR_PORT_SN to Socket2 to set source port Sport;
⑧ Send the command CMD_SET_PROTO_TYPE_SN to Socket2 to set the Socket to work in TCP mode;

Regarding the data structure, the process of sending data and receiving data can refer to TCP client mode.

9.2.7 DHCP

Before starting the following steps, the CH395 should be initialized first.

DHCP steps are as follows:

① Send the command CMD_DHCP_ENABLE parameter to 1 to start DHCP;
② Wait for CH395 to generate GINT_STAT_DHCP interrupt;
③ Wait for the GINT_STAT_DHCP interrupt and send the command CMD_GET_DHCP_STATUS to get the DHCP status, if the status code is 0 it means success, and the MCU can send the command CMD_GET_IP_INF to get the information of IP, MASK and so on. If the status code is 1, it means failure, there may be a problem in the connection between CH395 and DHCP Server, for example, the DHCP Server is not online, although CH395 notifies the MCU of the DHCP error through interrupts, the MCU still keeps retrying internally to find the DHCP Server, and the MCU can send the CMD_DHCP_ENABLE command with a parameter of 0 to stop DHCP. The MCU can send CMD_DHCP_ENABLE command with parameter 0 to stop DHCP.

9.2.8 About TCP MSS and Buffer

CH395 supports modifying TCPMSS, and the default TCPMSS size is 1460. Generally speaking, the larger the TCPMSS, the higher the communication speed and efficiency. Some principles should be followed when modifying TCPMSS and receive buffer.

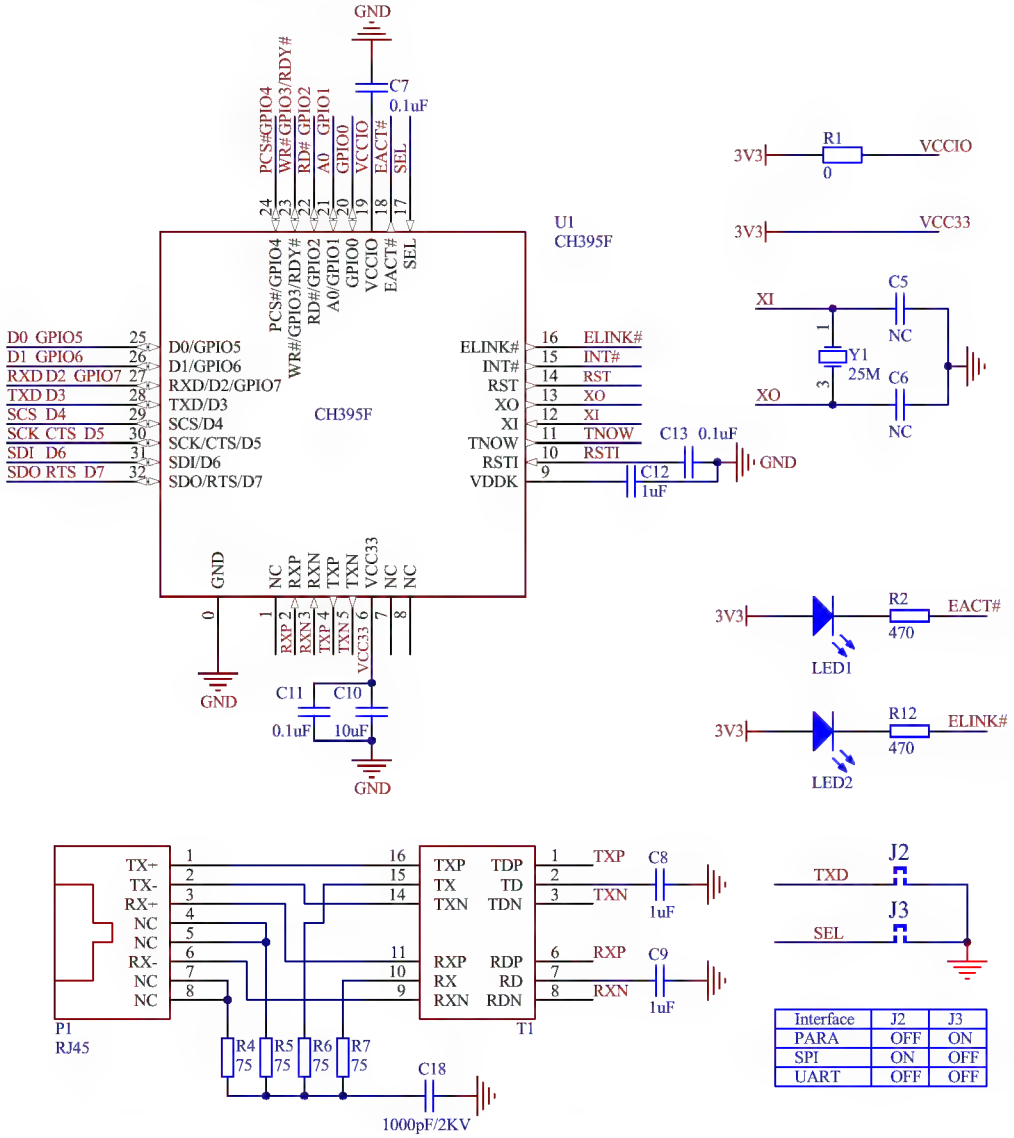
① (Recommended) The length of the receive buffer must not be less than 2 times the TCP MSS;
② (Must) The length of the receive buffer must not be less than the TCP MSS;
③ (Recommended) The length of the receive buffer must not be greater than 6 times the TCP MSS;
④ (Must) The send buffer size must not exceed 8KB.

When modifying TCPMSS, it should be as large as possible. If it is too small, it may lead to the increase of packet data on Ethernet and affect communication efficiency.

9.3 Reference Circuit

9.3.1 CH395F Reference Circuit

Figure 9-1 CH395F Hardware reference circuit



CH395F has built-in part of the oscillation capacitance of crystal Y1, C5 and C6 can be adjusted according to the crystal parameters. For Y1 with a load capacitance of 12pF, C5 and C6 are not required; for Y1 with a load capacitance of 20pF, C5 and C6 are recommended to be 15pF each.

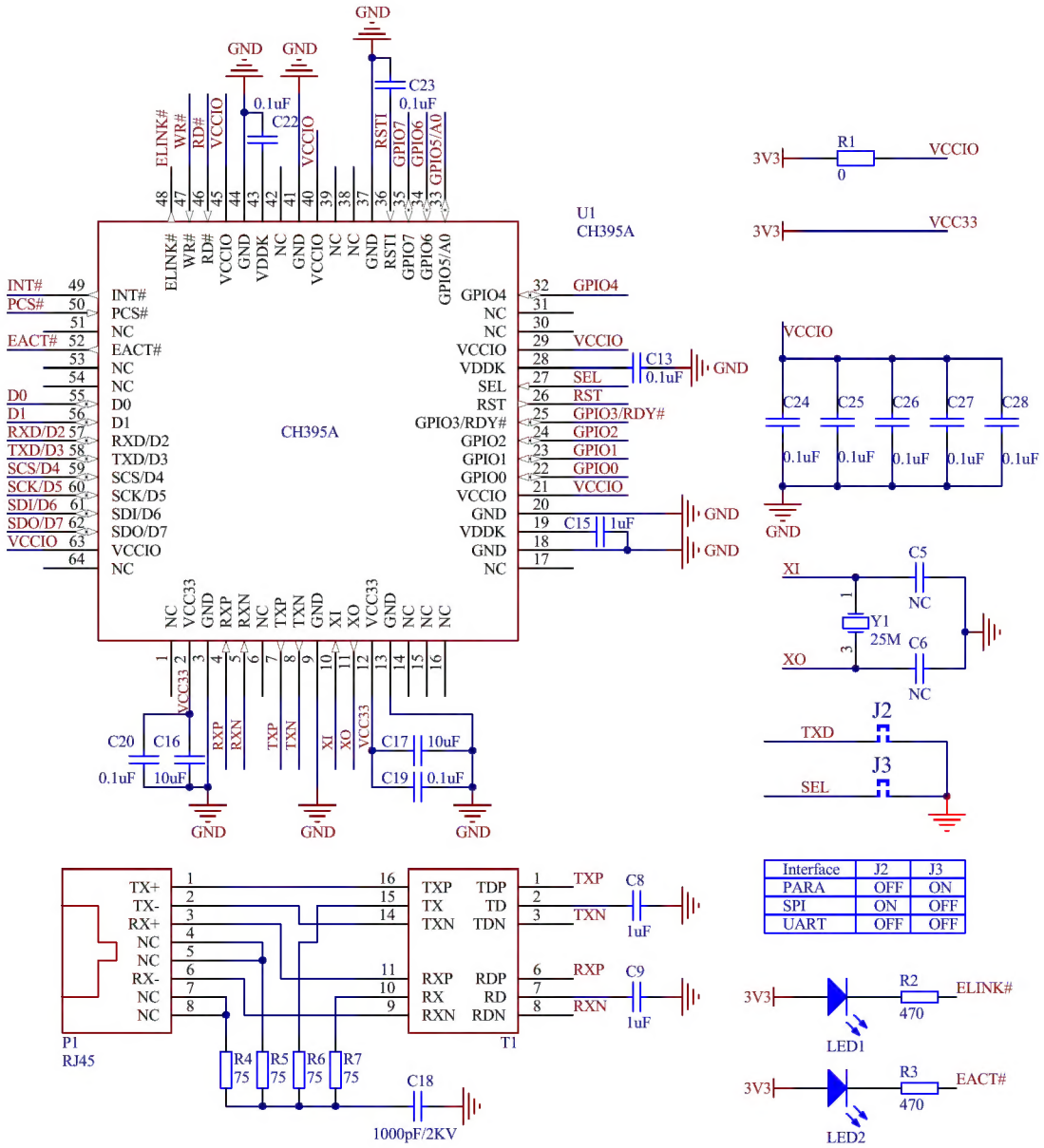
CH395F has built-in Ethernet 50Ω impedance matching resistor, do not connect 49.9Ω or 50Ω resistor externally, which is equivalent to voltage driver.

T1 is the ethernet network transformer, its center tap is grounded through capacitors C8 and C9 respectively, do not connect any power supply.

CH395F can select the desired communication interface through J2 and J3 jumper caps. Refer to section 6.1 for details on interface configuration.

9.3.2 CH395A Reference Circuit

Figure 9-2 CH395A Hardware reference circuit



CH395A has built-in partial oscillation capacitance of crystal Y1, and C5 and C6 can be adjusted according to crystal parameters. For Y1 with a load capacitance of 12pF, C5 and C6 are not needed; For Y1, C5 and C6 with a load capacitance of 20pF, 15pF each is recommended.

CH395A has built-in Ethernet 50Ω impedance matching resistor, so don't connect 49.9Ω or 50Ω resistor externally, which is equivalent to voltage driving.

T1 is an ethernet network transformer, and its center tap is grounded through capacitors C8 and C9 respectively, so don't connect any power supply.

C24, C25, C26, C27 and C28 should be placed close to the VCCIO pin respectively.

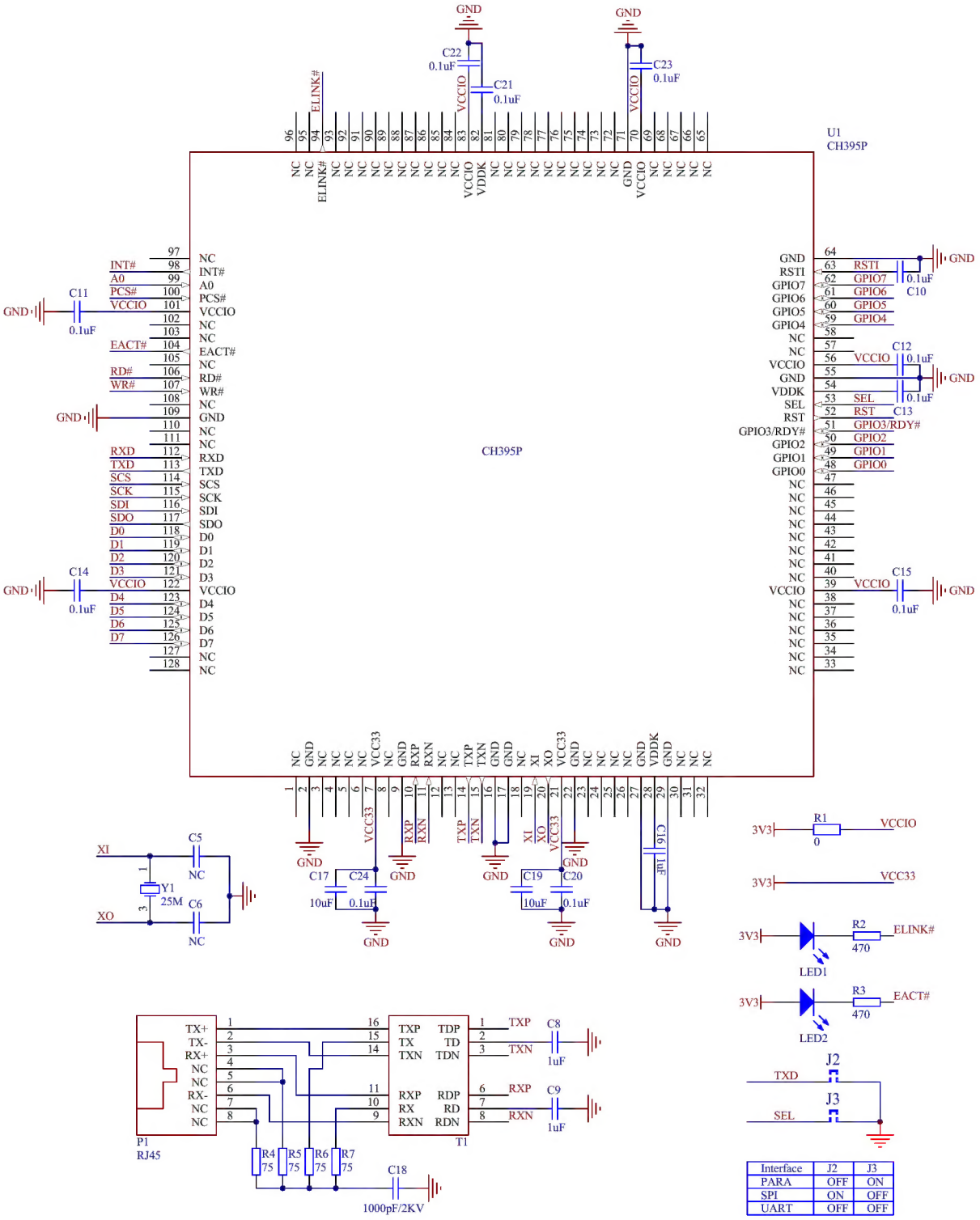
CH395A can select the required communication interface through J2 and J3 jumper caps. Refer to Section 6.1 for

details about interface configuration.

When upgrading from CH395Q to CH395A, although the pins are basically compatible, the connection of peripheral circuits such as network transformers may need to be adjusted. For specific changes, please refer to the relevant articles in WCH official forum or contact technical support.

9.3.3 CH395P Reference Circuit

Figure 9-2 CH395P Hardware reference circuit



Interface	J2	J3
PARA	OFF	ON
SPI	ON	OFF
UART	OFF	OFF

CH395P has built-in partial oscillation capacitance of crystal Y1, and C5 and C6 can be adjusted according to crystal parameters. For Y1 with a load capacitance of 12pF, C5 and C6 are not needed; For Y1, C5 and C6 with a load capacitance of 20pF, 15pF each is recommended.

CH395P has built-in Ethernet 50Ω impedance matching resistor, so don't connect 49.9Ω or 50Ω resistor

externally, which is equivalent to voltage driving.

T1 is an ethernet network transformer, and its center tap is grounded through capacitors C8 and C9 respectively, so don't connect any power supply.

CH395P can select the required communication interface through J2 and J3 jumper caps. Refer to Section 6.1 for details about interface configuration.

When upgrading from CH395L to CH395P, although the pins are basically compatible, the connection of peripheral circuits such as network transformers may need to be adjusted. For specific changes, please refer to the relevant articles in WCH official forum or contact technical support.